Gestion des activités pour les enseignants



Leonar Dupuis – FIN2

ETML - Vennes

88 heures

M. Gaël Sonney

(Nom et adresse du mandant)

Table des matières

[1 Spécifications 4](#_Toc168065292)

[1.1 Titre 4](#_Toc168065293)

[1.2 Description 4](#_Toc168065294)

[1.3 Matériel et logiciels à disposition 4](#_Toc168065295)

[1.4 Prérequis 4](#_Toc168065296)

[1.5 Cahier des charges 5](#_Toc168065297)

[1.5.1 Caractéristiques des utilisateurs et impacts 5](#_Toc168065298)

[1.5.2 Fonctionnalités requises (du point de vue de l’utilisateur) 5](#_Toc168065299)

[1.5.3 Contraintes 5](#_Toc168065300)

[1.5.4 Travail à réaliser par l'apprenti 5](#_Toc168065301)

[1.5.5 Si le temps le permet … 6](#_Toc168065302)

[1.5.6 Méthodes de validation des solutions 6](#_Toc168065303)

[1.6 Les points suivants seront évalués 6](#_Toc168065304)

[1.7 Validation et conditions de réussite 6](#_Toc168065305)

[2 Planification Initiale 7](#_Toc168065306)

[3 Analyse 16](#_Toc168065307)

[3.1 Méthodologie de travail 16](#_Toc168065308)

[3.2 Document d’analyse et conception 18](#_Toc168065309)

[3.2.1 Logo du site 18](#_Toc168065310)

[3.2.2 La maquette 19](#_Toc168065311)

[3.2.3 Schémas de base de données 28](#_Toc168065312)

[3.3 Conception des tests 30](#_Toc168065313)

[3.3.1 Test unitaire 30](#_Toc168065314)

[3.3.2 Test d’intégration 31](#_Toc168065315)

[3.3.3 Test de sécurité 31](#_Toc168065316)

[3.4 Planification détaillée 32](#_Toc168065317)

[4 Réalisation 32](#_Toc168065318)

[4.1 Dossier de Réalisation 32](#_Toc168065319)

[4.1.1 Environnement de projet 32](#_Toc168065320)

[4.1.2 Création et connexion à la base de données 34](#_Toc168065321)

[4.1.3 Gestion de l’authentification 36](#_Toc168065322)

[4.1.4 Détails du compte 41](#_Toc168065323)

[4.1.5 Opérations CRUD 44](#_Toc168065324)

[4.1.6 Liste des participants de l’activité 55](#_Toc168065325)

[4.1.7 Liste des activités & Inscription 57](#_Toc168065326)

[4.1.8 Consulter un profil utilisateur 59](#_Toc168065327)

[4.2 Modifications 60](#_Toc168065328)

[5 Tests 61](#_Toc168065329)

[5.1 Dossier des tests 61](#_Toc168065330)

[6 Conclusion 66](#_Toc168065331)

[6.1 Bilan des fonctionnalités demandées 66](#_Toc168065332)

[6.2 Bilan de la planification 66](#_Toc168065333)

[6.3 Bilan personnel 66](#_Toc168065334)

[7 Divers 66](#_Toc168065335)

[7.1 Journal de travail 66](#_Toc168065336)

[7.2 Bibliographie 66](#_Toc168065337)

[7.3 Webographie 66](#_Toc168065338)

[8 Annexes 66](#_Toc168065339)

[8.1 Résumé du rapport du TPI / version succincte de la doc. 66](#_Toc168065340)

[8.2 Glossaire 66](#_Toc168065341)

# Spécifications

## Titre

Gestion des activités pour les enseignants

## Description

À l'heure actuelle, l'incorporation d'outils numériques dans le domaine éducatif est devenue cruciale pour améliorer la gestion des diverses activités au sein des établissements scolaires.

Ce projet, qui est un travail pratique individuel, vise à développer une application web pour gérer les activités au sein d'une école, impliquant à la fois des élèves et des enseignants. Cette application permet aux enseignants d'organiser des activités et aux élèves de s'y inscrire. L'accès à l'application nécessite une authentification pour tous les utilisateurs.

Chaque activité, organisée par un enseignant, a un nombre maximum de participants, une date et un lieu définis. Les activités peuvent être de nature sportive telles que le tir à l'arc ou le tennis, ou des sorties culturelles. Les enseignants ont la possibilité de créer, modifier ou supprimer des activités, ainsi que de gérer la liste des participants. Ils peuvent également consulter la liste de leurs propres activités organisées. Les étudiants peuvent s'inscrire à des activités, consulter leurs inscriptions et se désinscrire si nécessaire.

Cette application veut donc répondre aux besoins actuels en matière de gestion des activités scolaires en exploitant les avantages des technologies web pour offrir une solution pratique, accessible et efficace.

## Matériel et logiciels à disposition

Pour démarrer ce projet, il est nécessaire d’avoir à disposition :

* Un PC standard de l’ETML (Windows 10)
* Visual Studio code avec un environnement PHP installé
* Docker
* La suite Microsoft Office pour les documents
* Un dépôt Git

## Prérequis

* Des connaissances en programmation PHP ainsi qu’en programmation orienté objet (POO) sont requises pour réaliser ce projet (références des modules à l’ETML : ICT 403, 404, 226, 120, 411, 133, 326).
* Des connaissances en implémentation et modélisation de bases de données relationnelles (références des modules à l’ETML : ICT 104, 105, 153).

## Cahier des charges

### Caractéristiques des utilisateurs et impacts

Si le projet est à déployer, le public cible pour ce projet serait les enseignants, les étudiants et les administrateurs scolaires.

Les conséquences que cela va avoir sur la conception sont, d’un point de vue ergonomique, que l’interface soit simple et intuitive. Les éléments bien visibles pour une navigation logique et une facilité d’utilisation.

Les couleurs du site doivent être harmonieuses tout en gardant un aspect professionnel, pour les enseignants et les administrateurs. Les couleurs vives pour les étudiants afin d’avoir un contraste suffisant pour la lisibilité et l’accessibilité.

### Fonctionnalités requises (du point de vue de l’utilisateur)

D’un point de vue utilisateur, les fonctionnalités présentes seront les suivantes :

* Gestion de l’authentification pour les enseignants et les élèves
* Opération CRUD sur les activités pour l’enseignant
  + Ajouter d’une activité
  + Modifier une activité
  + Supprimer une activité
  + Consulter les détails d’une activité
* Modifier la liste des participants (fonctionnalité enseignante)
* Consulter la liste des activités proposées
* S’inscrire pour une activité
* Consulter le profil utilisateur avec la liste des inscriptions

### Contraintes

Sécurité, backups, disponibilité, système utilisé, interfaces avec autres logiciels, etc.

### Travail à réaliser par l'apprenti

Une planification initiale à la fin du premier jour, ainsi qu’un rapport de projet

Une

### Si le temps le permet …

Un objectif complémentaire, si le temps le permet, serait de rendre le site web responsive**\***. En effet, il faut prendre en compte le fait que chaque utilisateur a une résolution d’écran différente et dans le cas d’un site non responsive, la navigation pourrait être désagréable et, dans le pire des cas, empêcher l’utilisateur d’accéder à certaines informations à cause des dimensions de certains éléments qui en cacheraient d’autres.

### Méthodes de validation des solutions

Comment les tests vont être entrepris, quels tests doivent être entrepris, etc.…

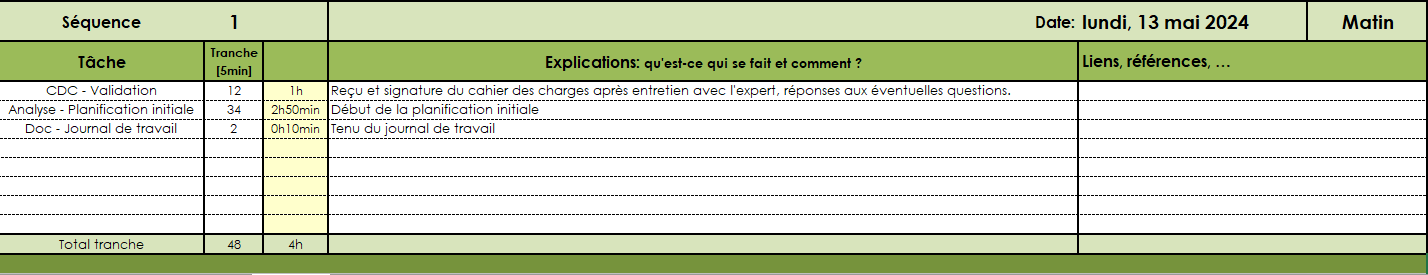
## Les points suivants seront évalués

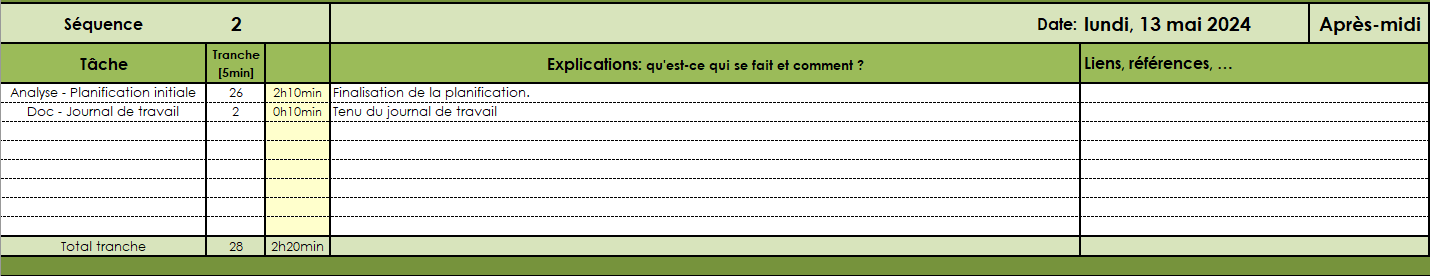
* Le rapport
* Les planifications (initiale et détaillée)
* Le journal de travail
* Le code et les commentaires
* Les documentations de mise en œuvre et d’utilisation

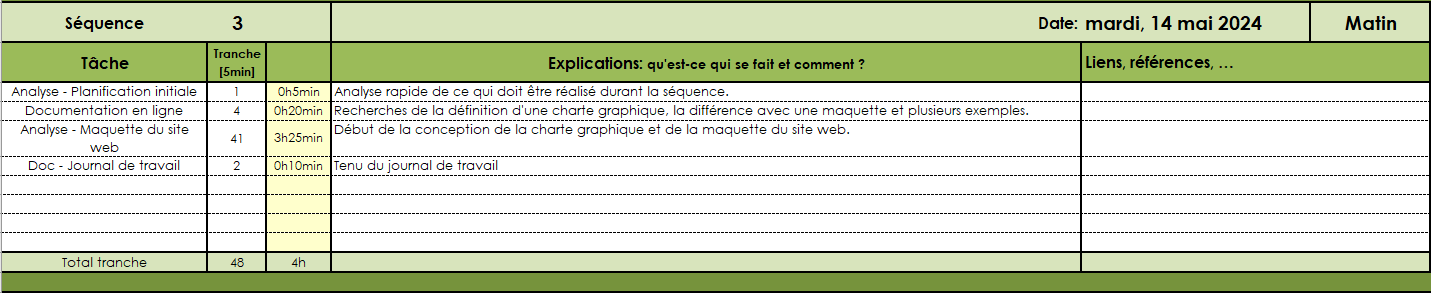
## Validation et conditions de réussite

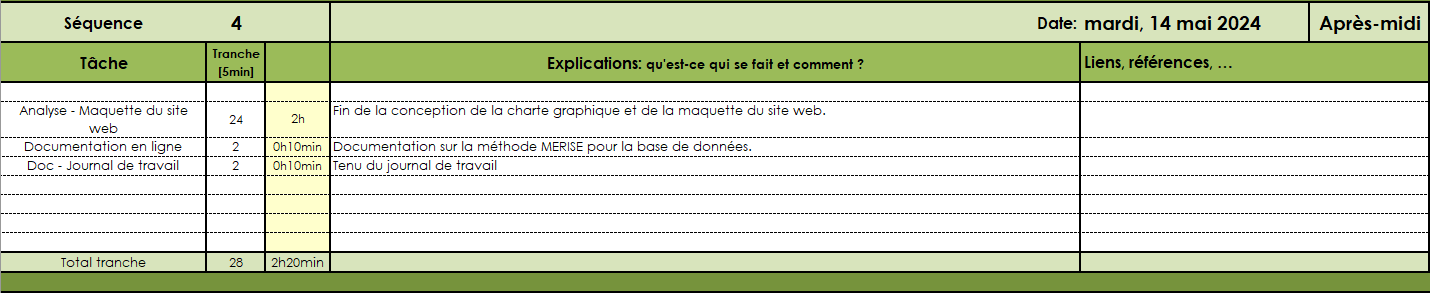
* Compréhension du travail
* Possibilité de transmettre le travail à une personne extérieure pour le terminer, le corriger ou le compléter
* Etat de fonctionnement du produit livré

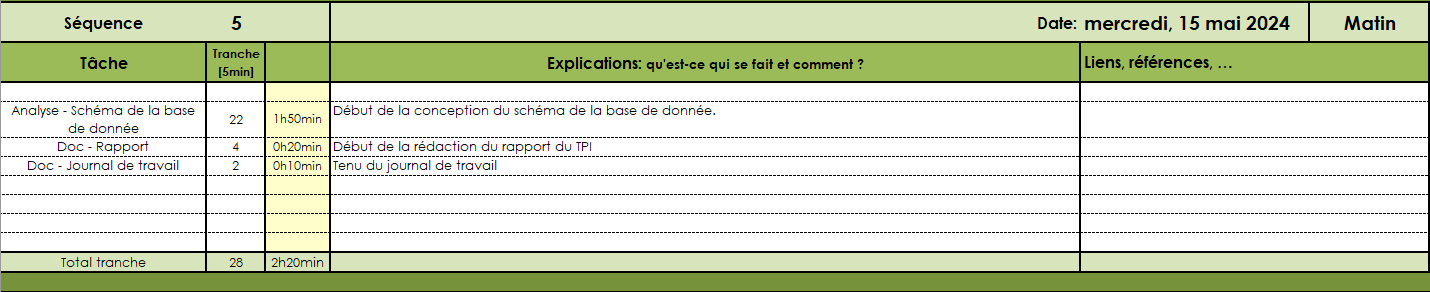
# Planification Initiale

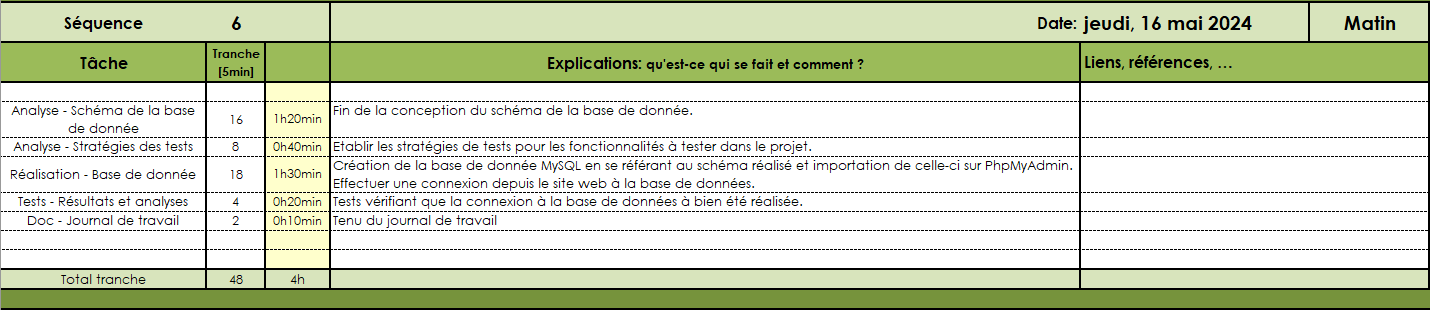


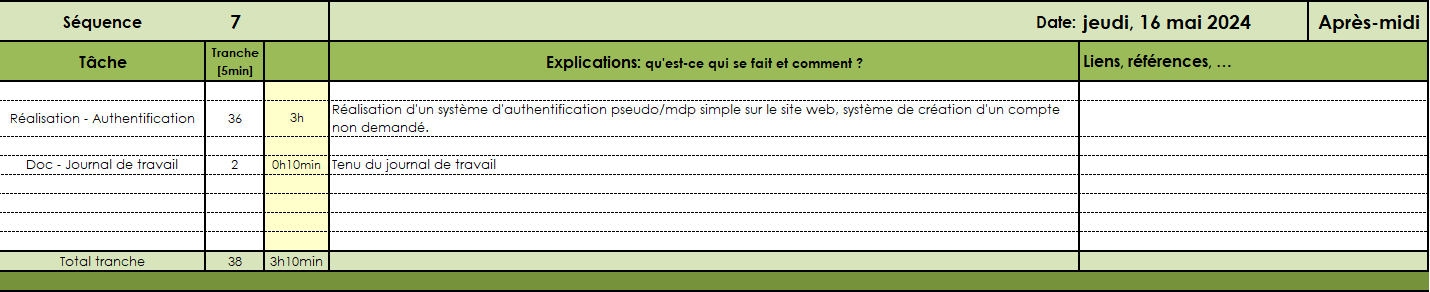


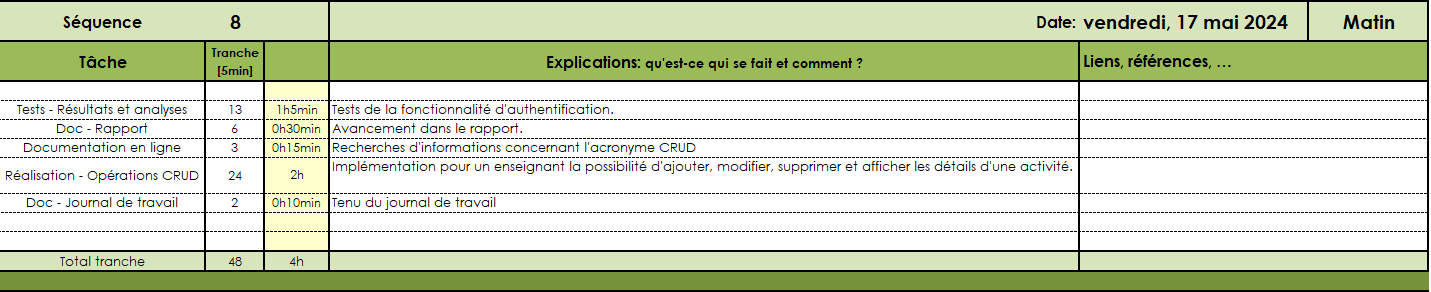


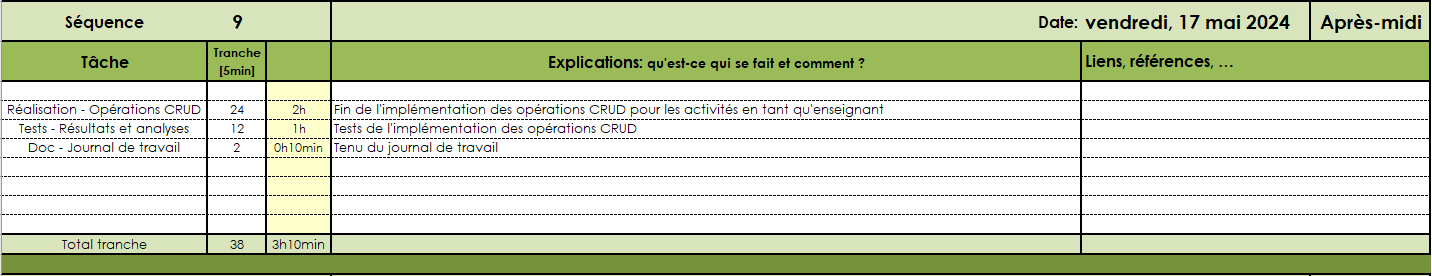


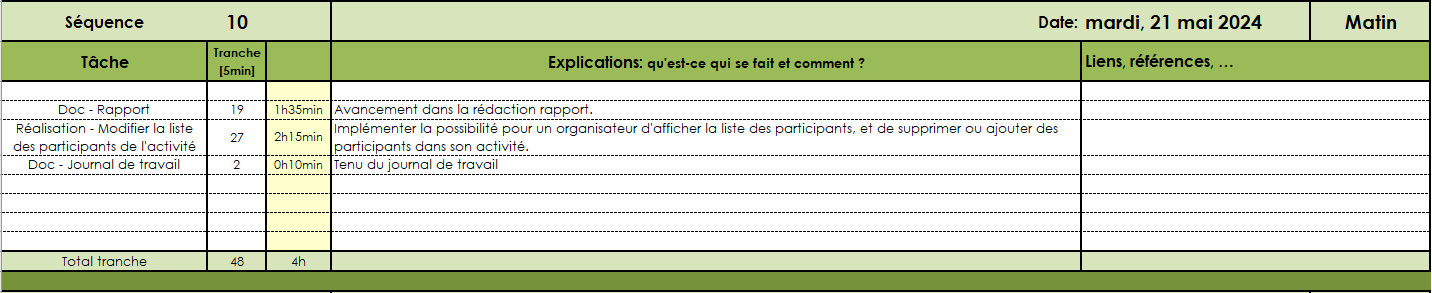


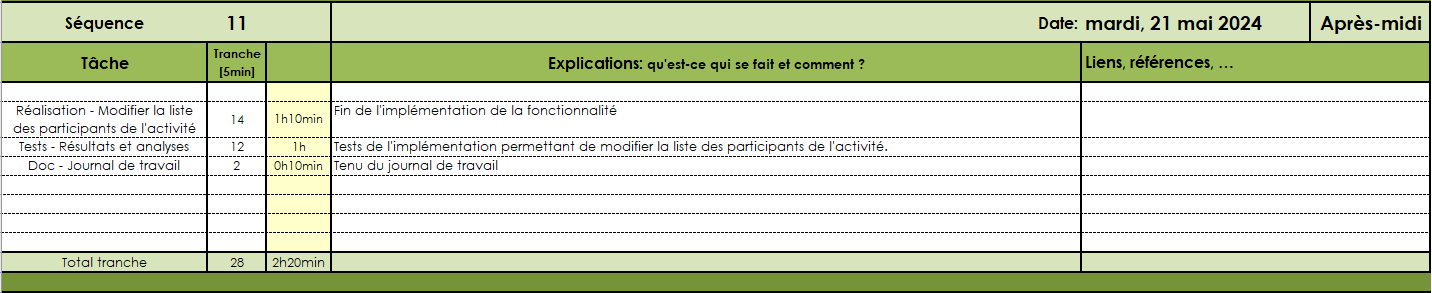


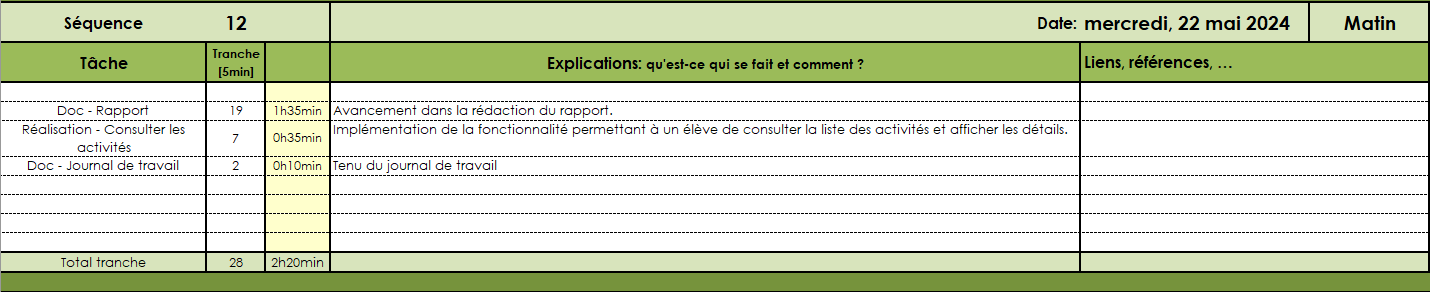


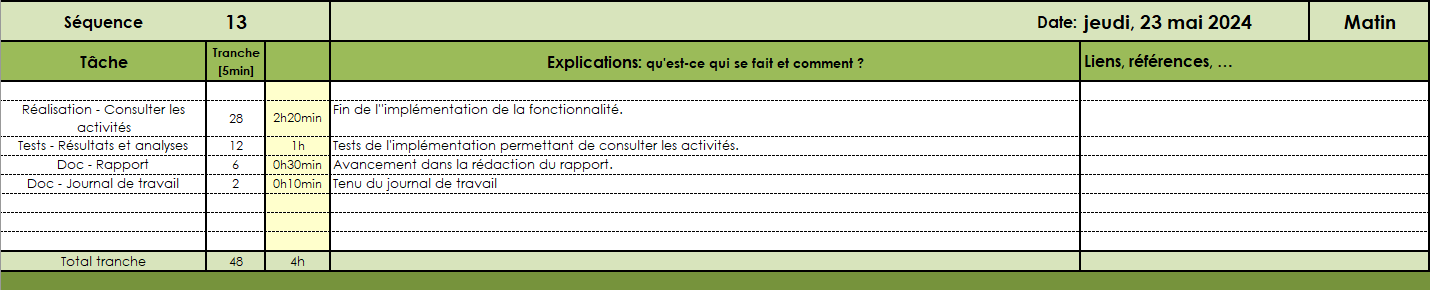


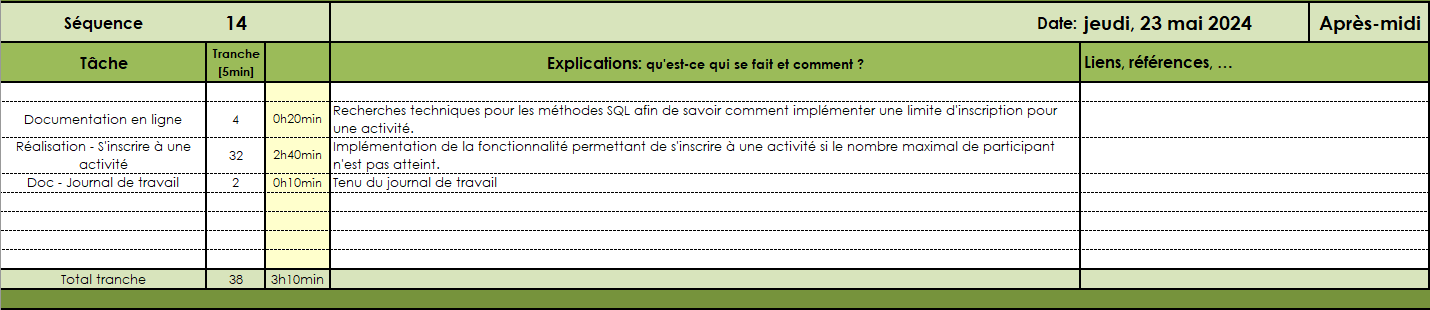


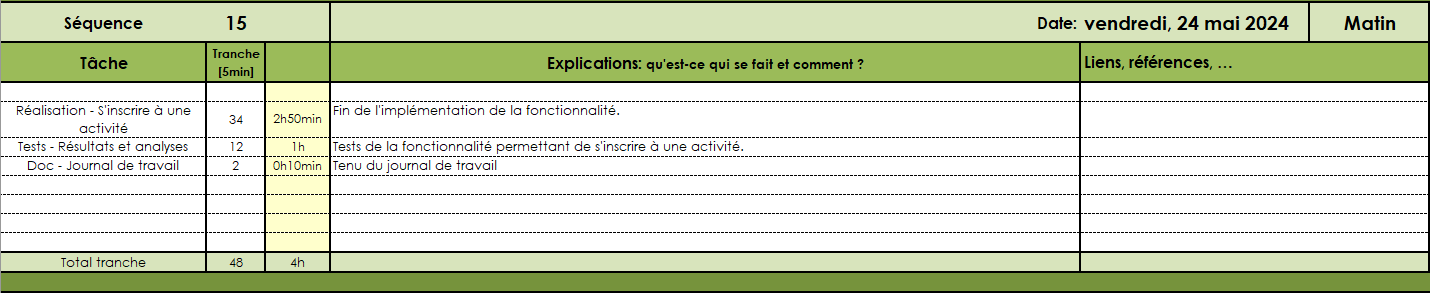


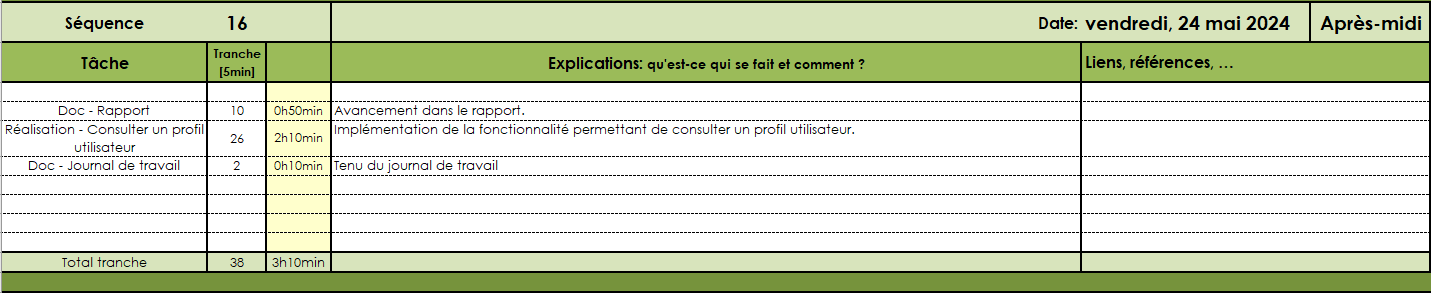


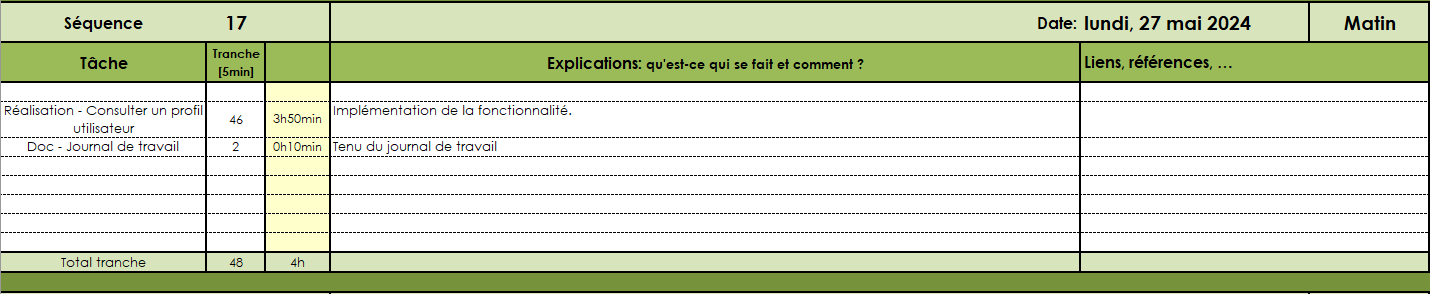


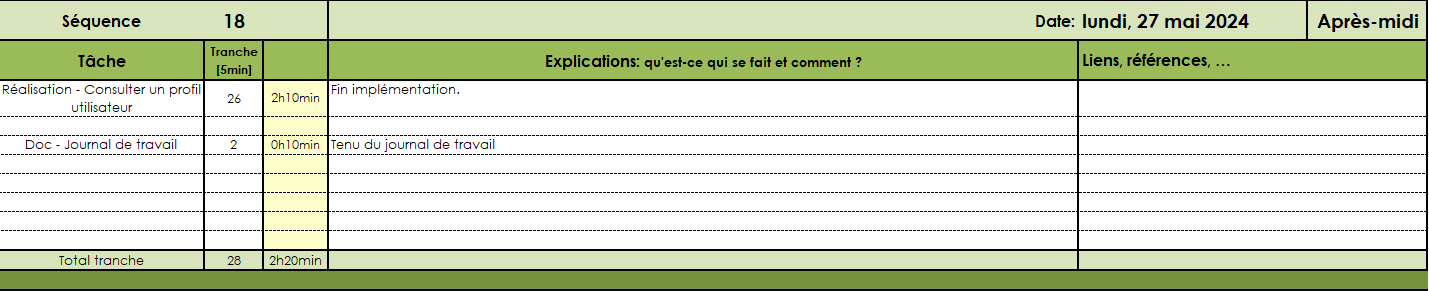


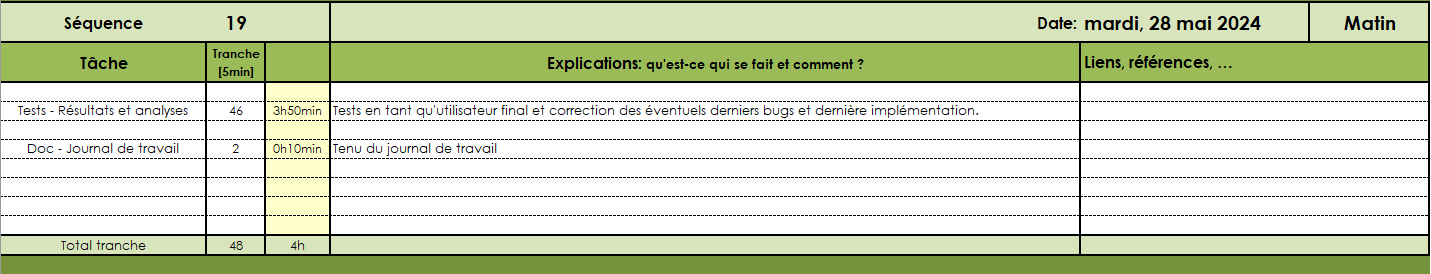


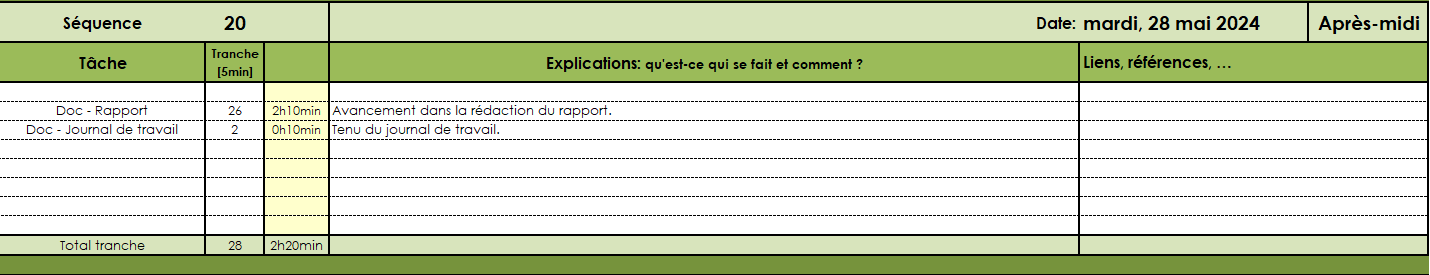


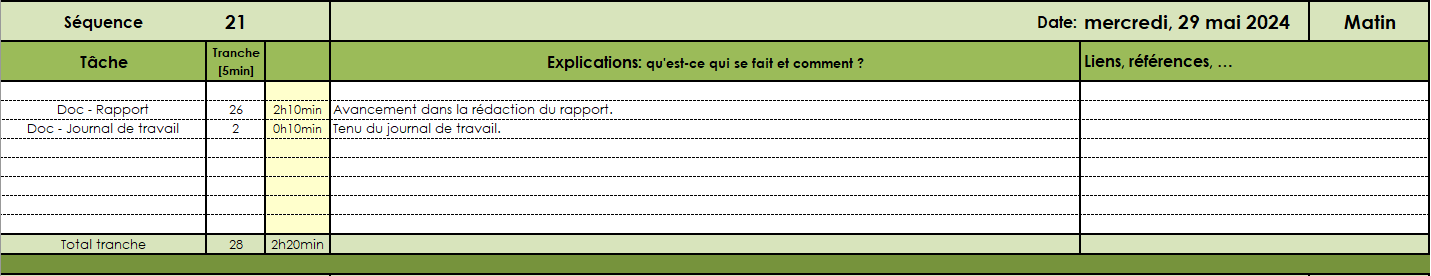


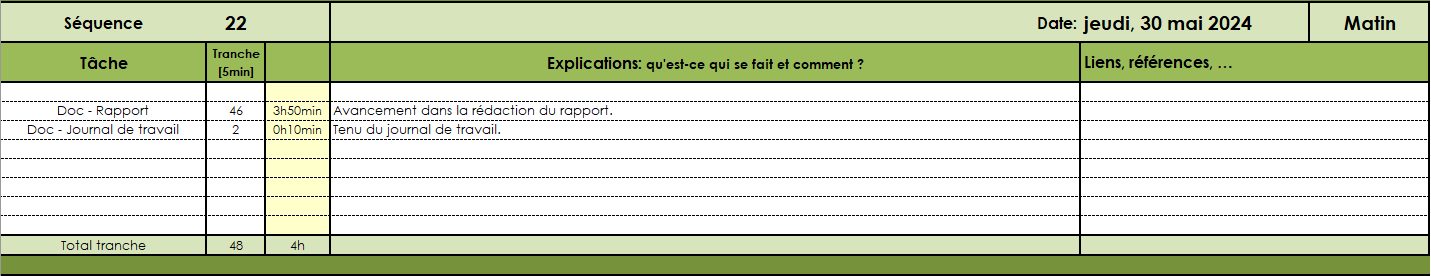


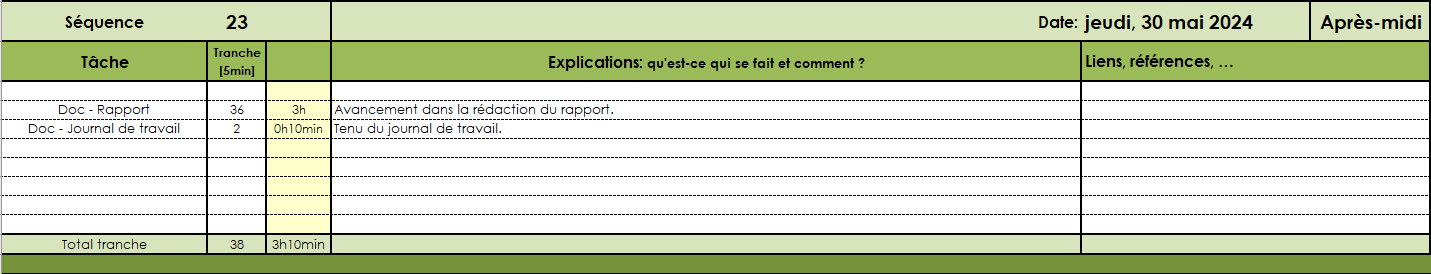


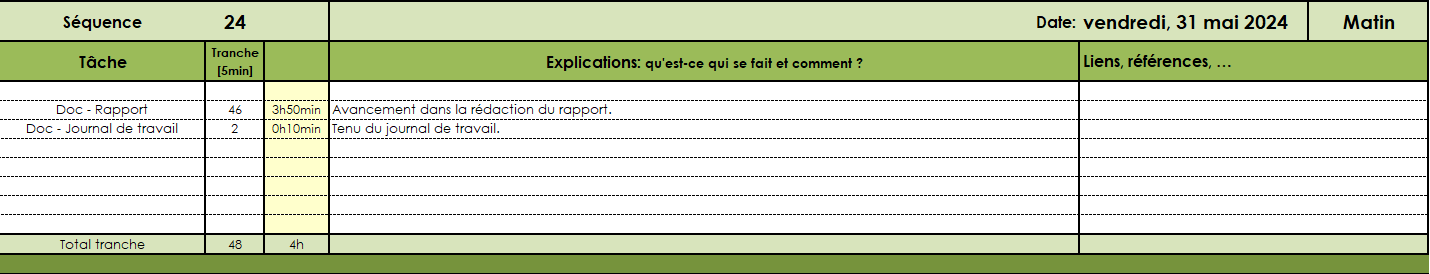


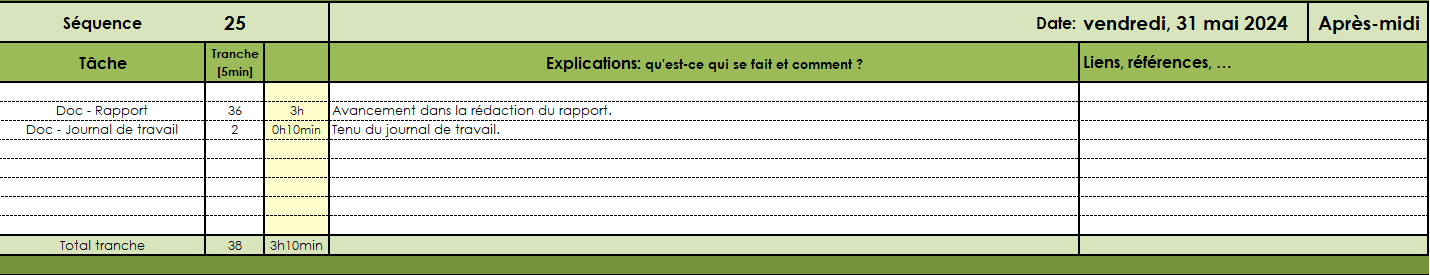


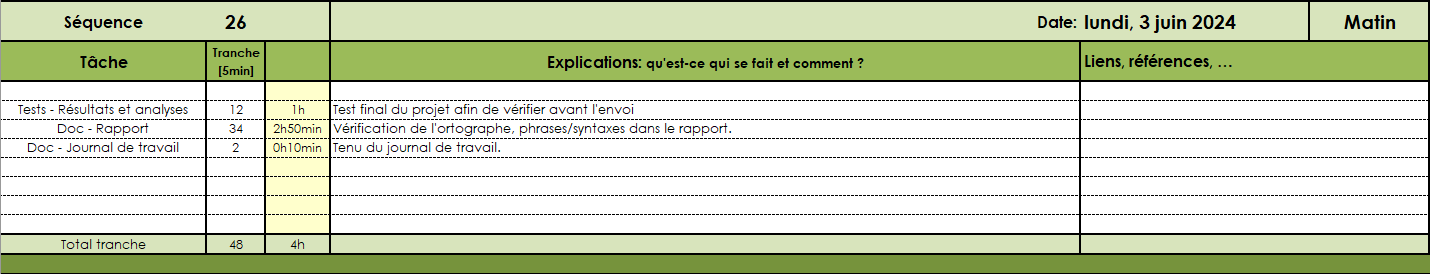


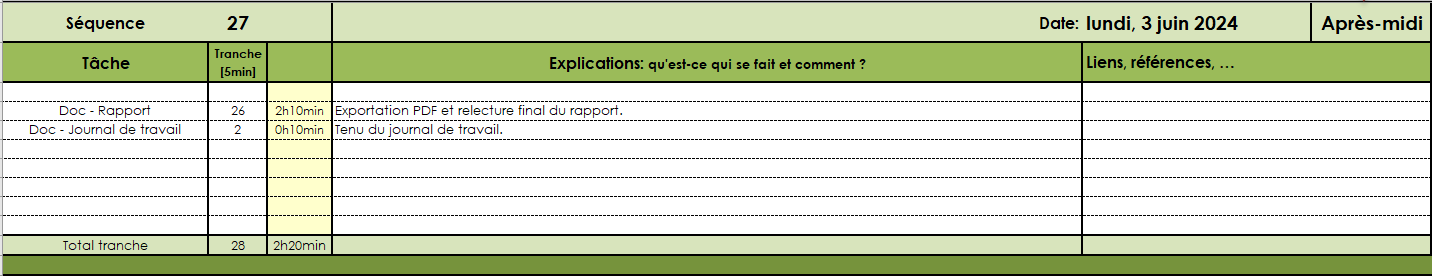












# Analyse

## Méthodologie de travail

Pour ce projet, j'ai délibérément opté pour la méthode Waterfall en raison de sa structure linéaire, offrant ainsi une vision précise et détaillée à chaque étape du développement. Avec des exigences clairement énoncées dans le cahier des charges et un temps de projet relativement court (moins d'un mois), la méthode Waterfall représente pour moi la meilleure approche pour garantir une planification minutieuse et une exécution efficace.

Les tâches créées dans mon journal de travail sont basées sur la méthodologie ci-contre, détaillant chaque étape du processus, les décisions prises, les problèmes rencontrés et les solutions apportées, assurant ainsi une traçabilité complète et une transparence tout au long du projet.

|  |  |
| --- | --- |
| **Etapes** | **Approche** |
| Analyse des besoins | * Une planification initiale à la fin du premier jour. * Un rapport de projet complet fidèle aux critères d’évaluation du travail pratique individuel (TPI) à livrer 2 fois par semaine sous format pdf. * Une journalisation détaillée à livrer 2 fois par semaine sous format pdf. |
| Conception | * Pour réaliser les maquettes web, j’aurai besoin de Figma. * Les schémas de bases de données seront réalisés à l’aide du logiciel Looping. * Le logo sera réaliser grâce au modèle DALL-E, une extension de ChatGPT 4. |
| Implémentation | * Je vais utiliser des langages de programmation comme HTML, CSS, PHP. * Le code source sera organisé avec un contrôle de version à l’aide de Git pour assurer un bon historique de version et une sauvegarde de données. * Un SSD sera utilisé pour bénéficier d’un backup\*. * Les modifications et ajouts au code seront gérés à l’aide de push sur la branche principale git. |

|  |  |
| --- | --- |
| **Etapes** | **Approche** |
| Tests | * Les tests incluront des tests unitaires pour chaque fonctionnalité, des tests d'intégration pour vérifier l'interaction entre les composants, et des tests fonctionnels pour valider le comportement global de l'application. * Des tests continus seront faits. Chaque fonctionnalité implémentée sera directement testée. * Plusieurs cas de tests seront effectués pour chaque fonctionnalité. * Une fonctionnalité sera validée une fois que tous les cas de tests seront effectués. |
| Déploiement/Livraison | * Le lundi 3 juin, le projet sera livré dans sa version final avec, selon la volonté des experts, la possibilité d’avoir une impression papier. |
| Maintenance  (Amélioration Post-Défense) | * Possibilité d’améliorer l’application après le recueil des retours du chef de projet et des experts. * Noter les points de critique constructifs afin d’améliorer la réalisation d’un projet comme celui-ci dans le futur. |

## Document d’analyse et conception

### Logo du site

Tout d’abord, avant de m’attaquer au design de mon site web, j’ai utilisé **ChatGPT** 4 avec le modèle DALL-E intégré pour me générer un logo attrayant, rappelant les activités sportives. Le **prompt** était le suivant :

*« Génère-moi une image de logo d’un site web pour gérer les activités sportives d’un site (organisateur à élèves). Je veux un rendu énergique. »* Le premier résultat était le suivant :



Figure 1 - Premier logo généré

Le résultat m’a plu, l’**IA** a bien intégré l’adjectif « énergique » que je lui avais demandé, mais le fond n’était pas blanc et aucune personne n’était visible sur le logo, ce qui me dérangeait. Alors je lui demandé de me refaire une proposition avec cette fois-ci un fond blanc et une personne, je suis arrivé à ce résultat au bout de la quatrième génération :



Figure 2 - Logo final

Cette génération m’a finalement convaincu. Le fond blanc et l’esthétique colorée du logo me laissent une large palette de possibilités pour la réalisation de la maquette du site, en restant dans le thème d’un « site de gestion d’activités ».

### La maquette

Pour réaliser la maquette de mon site web, j’ai utilisé **Figma** et me suis concentré essentiellement aux pages de fonctionnalités, tout d’abord d’une vue d’utilisateur « élève » puis d’une vue utilisateur « enseignant » qui a des droits supplémentaires comparé aux élèves. M’étant beaucoup inspiré du logo, l’esthétique du site tourne autour de la couleur verte afin d’avoir une bonne cohérence graphique. La maquette de ce site web a été réalisée en me fixant comme objectif de respecter les critères UX : une interface cohérente, crédible, simple, non chargée inutilement et interactive.

J’ai choisi de réaliser les maquettes avant de concevoir les schémas de la base de données pour plusieurs raisons. Tout d’abord, les maquettes permettent de visualiser l’interface utilisateur finale et d’identifier les éléments nécessaires sur un site, ce qui aide à définir les entités et les relations dans la base de données.

Ayant une approche plus visuelle dans mon travail, commencer par les maquettes m’a personnellement permis d’identifier plus facilement les besoins fonctionnels et les interactions des utilisateurs.



Figure 3 - Page d'accueil



Figure 4 - Page de connexion



Figure 5 - Connexion échouée

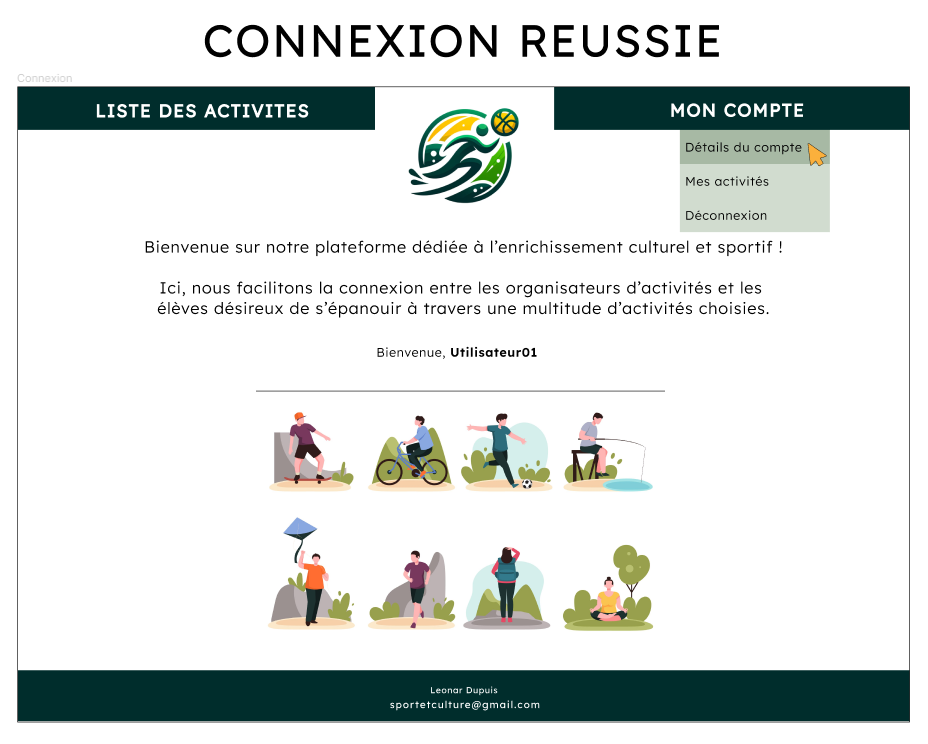


Figure 6 - Connexion réussie



Figure 7 - Détails du compte

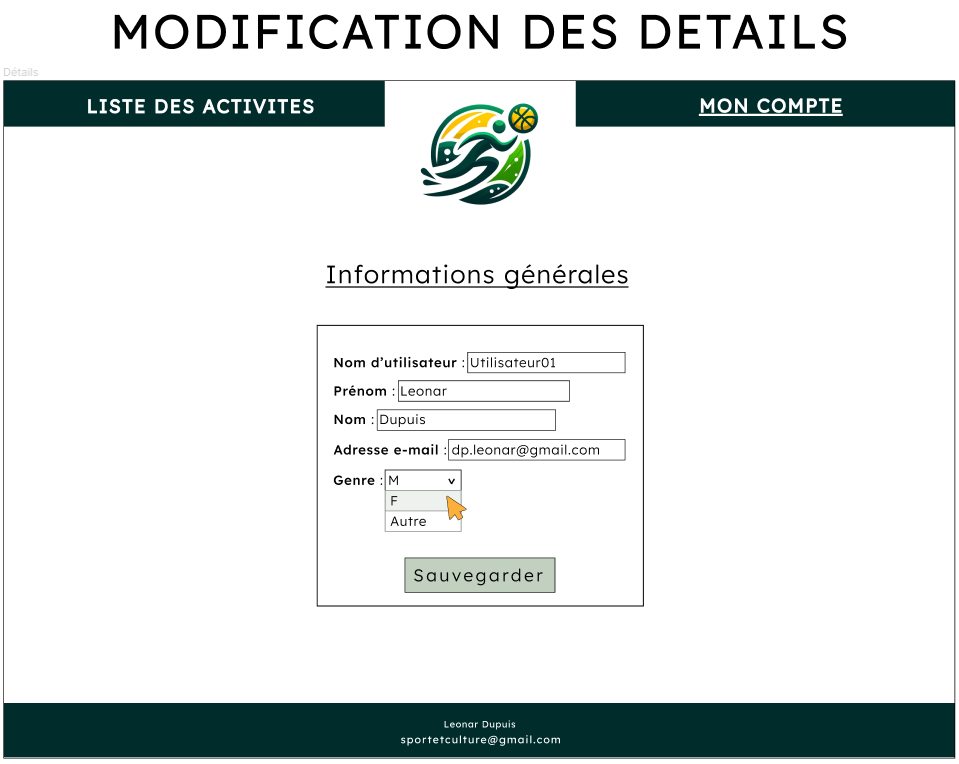


Figure 8 - Modification des détails



Figure 9 - Mes activités (vue élève)



Figure 10 - Mes activités (vue organisateur)



Figure 11 - Modifier une activité (organisateur)



Figure 12 - Créer une activité (organisateur)



Figure 13 - Liste des activités



Figure 14 - inscription à une activité réussie (élève)



Figure 15 - inscription à une activité réussie (élève)



Figure 16 - Détails & Liste participants



Figure 17 - Profil élève



Figure 18 - Profil enseignant

### Schémas de base de données

Pour la modélisation de ma base de données, j’ai utilisé le logiciel Looping qui est très facile à prendre en main. Les schémas suivants ont été réalisés en prenant en compte la convention de nommage standards de l’ETML.

Schéma **MCD** :

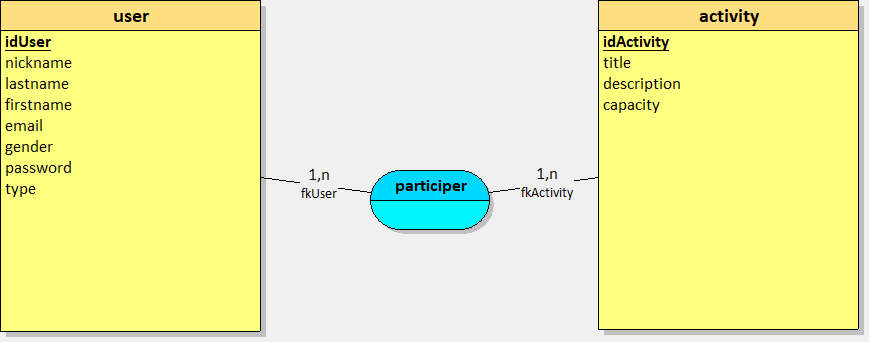


Figure 19 - Schéma MCD

Schéma **MLD** :

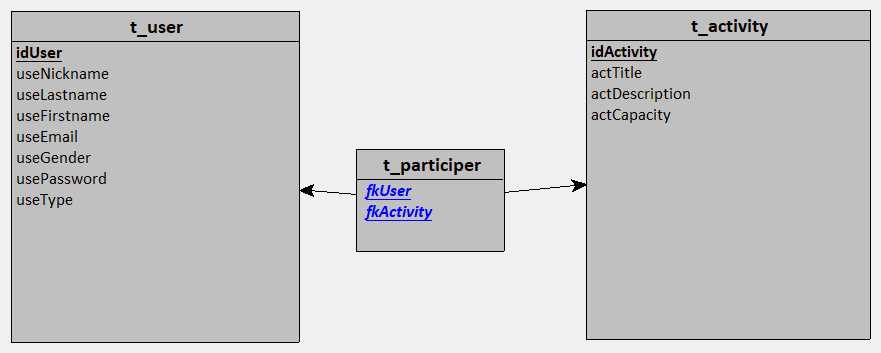


Figure 20 - Schéma MLD

Schéma **MPD** :

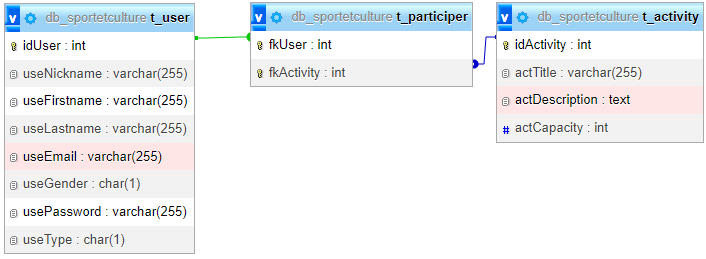


Figure 21 - Schéma MPD

La conceptualisation de ma base de données a été basé sur la méthode MERISE, qui encourage à utiliser les formes normales afin d’assurer l’intégrité et l’efficacité des données. J’ai donc créé mes schémas en m’assurant de respecter les 3 premières formes normales :

* La **première** forme normale veut que chaque table ait une clé primaire unique, (exemple : ‘idUser’). Chaque valeur est atomique, ce qui veut dire que chaque entrée dans une table est une valeur simple et indivisible (comme « useLastname »).
* La **deuxième** veut aucune dépendance partielle. Concrètement, en prenant l’exemple de la table t\_activity, le titre, la description et la capacité de l’activité dépendent entièrement de idActivity.
* La **troisième** forme veut qu’aucun attribut d’une table n’ait de dépendance envers un autre attribut non-clé. C’est-à-dire qu’il n’y a aucune dépendance transitive. Par exemple, dans la table t\_user, le nom, le prénom et l’email de l’utilisateur ne dépendent pas d’autres champs non-clés.

## Conception des tests

Les stratégies de test permettent de détecter et de corriger les erreurs afin d'assurer le bon fonctionnement des fonctionnalités et offrir une expérience utilisateur optimale. En utilisant des tests unitaires, d'intégration, fonctionnels et de sécurité, on peut identifier les problèmes avant qu'ils n'affectent les utilisateurs finaux.

Les tests sont planifiés après chaque implémentation de fonctionnalité. Cela permet de maintenir une intégrité globale de l’application, permettant d’éviter les risques de conflits et maintenir une bonne cohésion dans l’ensemble du système.

En fonction du temps disponible, un test en tant qu’utilisateur final sera inclus afin de garantir une validation complète de l’application.

### Test unitaire

Les tests unitaires visent à vérifier le bon fonctionnement d’éléments précis dans le code, telles que les fonctions et les méthodes, de manière isolée.

La démarche de mes tests unitaires est la suivante :

* **Sélection des fonctions**

J’identifie les fonctions à tester dans mon code.

* **Cas de test**

Pour chaque fonction à tester, je définis les entrées que je vais faire et les résultats attendus.

* **Exécution manuelle**

J’exécute manuellement ces fonctions avec les différentes entrées et je compare les résultats obtenus avec les résultats attendus. Si le travail obtenu n’est pas satisfaisant, je revisionne mon code avant de réeffectuer la procédure de test.

### Test d’intégration

Les tests d’intégration vérifient que les différents modules du code fonctionnent bien ensemble.

La démarche de mes tests d’intégrations est la suivante :

* **Scénarios d’intégration**

J’identifie une fonctionnalité qui interagit avec différents modules de mon site web.

* **Cas de test**

Je définis un scénario de test, où je suis par exemple un utilisateur qui va provoquer l’interaction de plusieurs modules entre eux en utilisant une fonctionnalité. Je définis plusieurs scénarios de tests, cherchant plusieurs chemins d’exécution possibles.

* **Exécution manuelle**

J’exécute ce scénario en comparant les résultats obtenus avec les résultats attendus.

### Test de sécurité

Les tests de sécurité permettent d’identifier les vulnérabilités potentielles dans une application web. C’est un point très sensible, surtout quand notre site contient du PHP.

La démarche de mes tests de sécurité est la suivante :

* **Vulnérabilités connues**

J’identifie les types courants de vulnérabilités (ex : injection SQL).

* **Cas de test**

Je définis un ou plusieurs tests pour chaque type de vulnérabilités.

* **Exécution manuelle**

Je tente d’exploiter ces vulnérabilités en utilisant des entrées malveillantes, par exemples dans des formulaires ou des points de saisie.

## Planification détaillée

A ce stade, après l’analyse complète du projet, un planning détaillé et complet (avec tâches, sous-tâches, dépendances, durée, …) peut être finalisé.

Le planning détaillé doit s’inscrire dans le planning initial. Il faut que l’on puisse situer cette planification détaillée par rapport à la planification initiale.

# Réalisation

## Dossier de Réalisation

### Environnement de projet

Tout d’abord, il est impératif d’avoir un bon environnement de travail pour pouvoir commencer ce projet. Le projet étant versionné sur GitHub, un readme**\*** expliquant comment l’environnement a été configuré se trouve à la racine de l’arborescence du site.

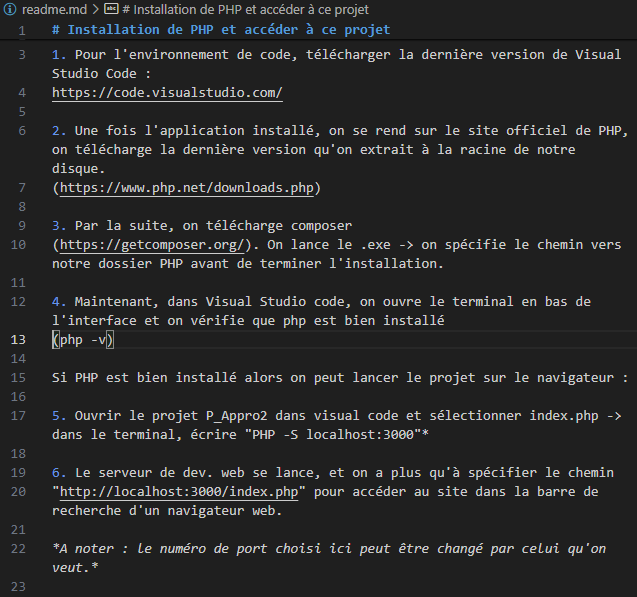


Figure 22 - readme p.1

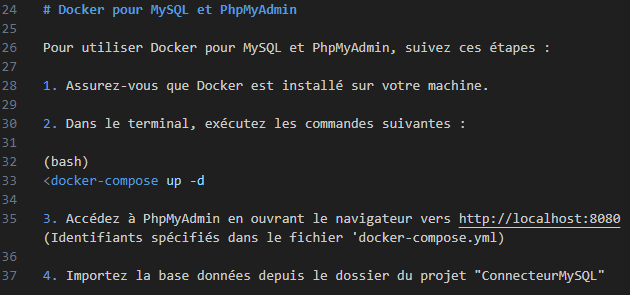


Figure 23 - readme p.2

En ce qui concerne l’organisation des fichiers en tant que tel pour le projet, une structure **MVC** a été mis en place. Il s’agit d’un modèle d’architecture très répandu dans la conceptualisation de logiciels avec interfaces graphiques. Dans ce projet, la logique est la suivante :

* Les fichiers qui sont montrés/vus par des utilisateurs sont placés dans le répertoire « views ».
* Les fichiers gérant des conditions, des erreurs avant d’être envoyés dans une méthode vont passer dans le répertoire « controllers ». Il gère les interactions entre le modèle et la vue.
* Les fichiers qui interagissent avec la base de données via des fonctions, sont dans le répertoire « models ».

|  |  |
| --- | --- |
| Figure 24 - models / controllers | Figure 25 – resources > views |

### Création et connexion à la base de données

Le début de ce projet a consisté pour moi à créer, dans mon dossier ConnecteurMySQL, ma base de données « db\_sportetculture » puis de créer mes tables en me basant sur le modèle créé avec l’application Looping :

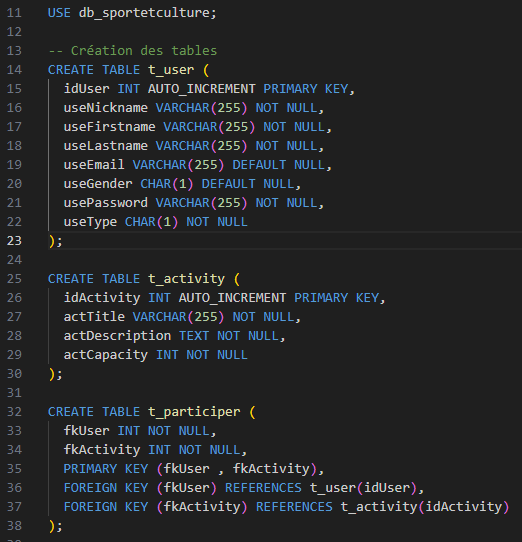


Figure 26 - Création des tables SQL

Maintenant que la base de données est prête, on se rend sur PhpMyAdmin en utilisant les champs par défaut : **root**, **root.** Puison crée la base de données, en laissant par défaut le choix de caractères :

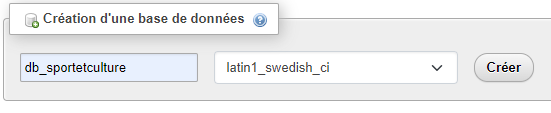


Figure 27 - Création de la base de données

On importe ensuite notre code de création des tables, et on vérifie que l’importation ait bien réussie :

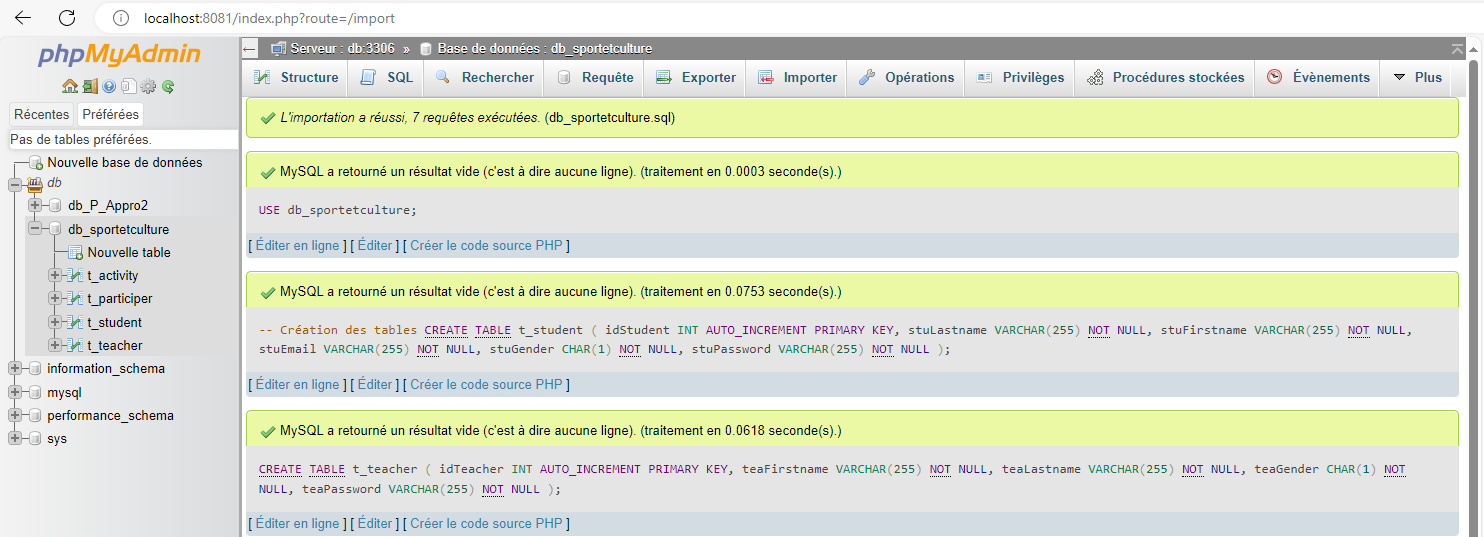


Figure 28 - Importation réussie

Une fois importée, on établit la connexion entre notre site web et la base de données. Dans la partie « models » de mon arborescence web, là où les données sont manipulées, on crée le fichier database.php.

Dans notre classe Database, dans la fonction \_\_construct(), on se connecte via PDO (extension définissant l'interface pour se connecter à une base de données avec PHP) en spécifiant les informations à rentrer sur mon PhpMyAdmin comme le mot de passe. Si la connexion est établie, un message «db connecté» apparaît sur le site, sinon, un message d’erreur survient.

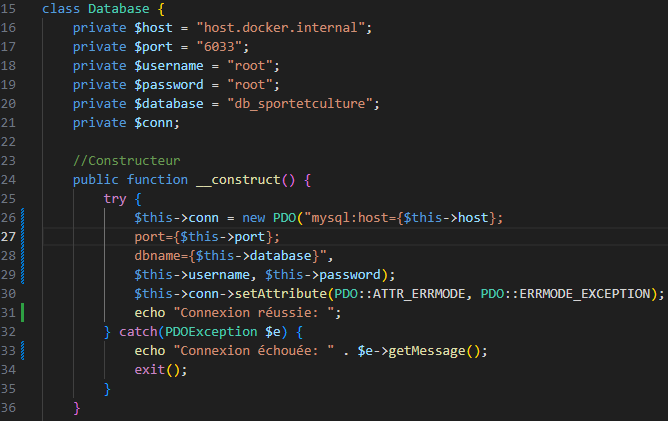


Figure 29 - Connexion à la base de données

La connexion ici marche donc sans problème, grâce au message visible à l’accueil :



Figure 30 - Connexion réussie

### Gestion de l’authentification

Sur le site web, un système de connexion a été mis en place avec la possibilité de s’inscrire bien que cette fonctionnalité soit optionnelle. Elle est disponible pour l’utilisateur en haut à droite dans ma barre de navigation.



Figure 31 - Barre de navigation (visiteur)

Arrivé sur la page, l’utilisateur doit remplir un formulaire basique avec un mot de passe à confirmer.

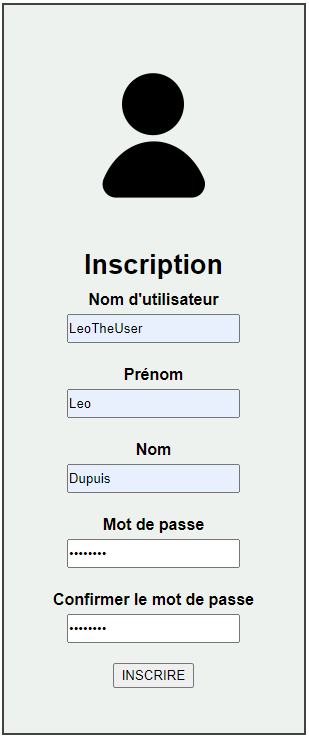


Figure 32 - Inscription

Cette page est présente dans le répertoire « views » dans mon site web, c’est-à-dire qu’il s’agit de la partie visible pour l’utilisateur, mais la page de vérification des champs (gestion d’erreurs), elle, va être dans ma partie controllers et va vérifier que l’utilisateur n’a pas sauter un champ :



Figure 33 - Vérification des champs

Si l’utilisateur a bien rempli tous les champs, alors le formulaire va passer dans la partie « models », dans une fonction « **registerUser** » qui contient comme paramètres les champs du formulaire à enregistrer dans la base de données. Il s’agit d’une simple requête d’insertion qui hash le mot de passe de l’utilisateur dans la base de données.

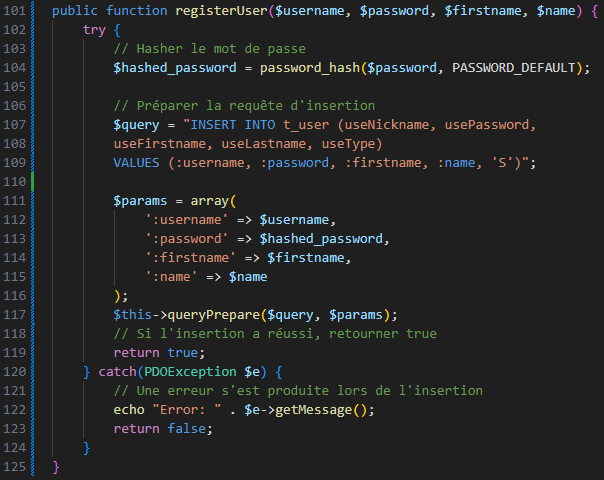


Figure 34 - Fonction "RegisterUser" – Hasher

On constate par la suite que les informations de l’utilisateur ne sont pas lisibles dans la base de données.



Figure 35 - Mot de passe haché

Dans la méthode précédente, on peut voir qu’une méthode « queryPrepare » est appelée. Celle-ci se composte en trois étapes :

* Préparer une requête SQL
* L’exécuter avec des paramètres
* Gérer les erreurs.

C’est une bonne pratique qui aide à prévenir les erreurs, ou encore les injections SQL.

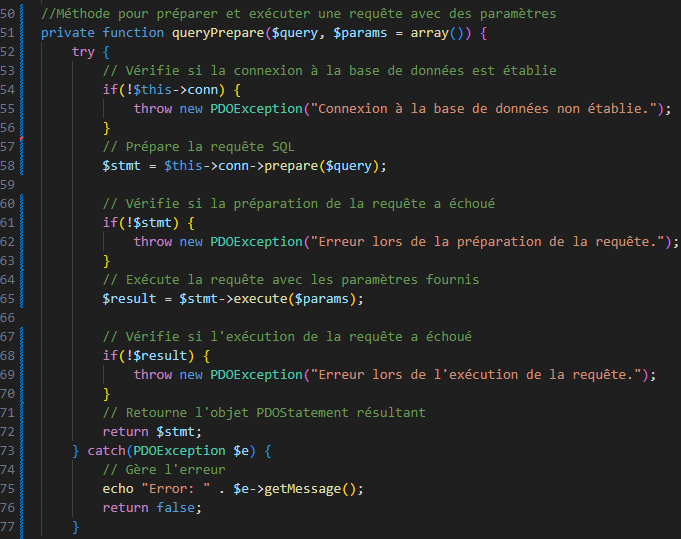


Figure 36 - QueryPrepare

Pour revenir à l’utilisateur, on peut voir que celui-ci a bien été enregistré dans la base de données :

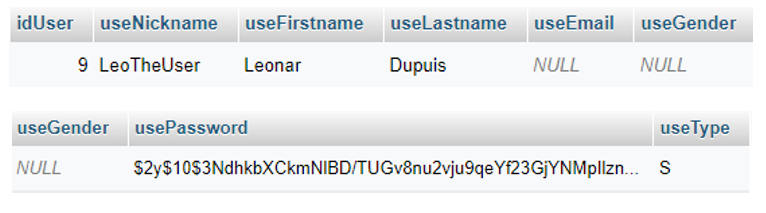


Figure 37 - Utilisateur inscrit

L’utilisateur peut maintenant se connecter.



Figure 38 - Connexion

Malgré tout, si les informations ne correspondent pas avec le bon nom d’utilisateur et son mot de passe, un message survient.

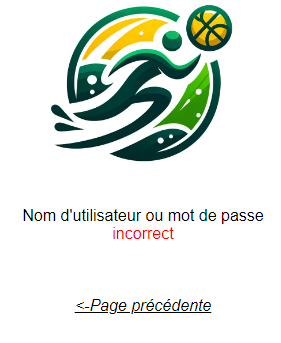


Figure 39 - Vérification connexion

La méthode utilisée ici vérifie juste que le nom d’utilisateur est le mot de passe sont corrects. Elle créée une requête qui, une fois préparée et exécutée, va être utilisée pour comparer le mot de passe fourni avec le mot de passe stocké (après le déchiffrement). Si le résultat est correct, les informations seront stockées dans la session, sinon, retourne false.

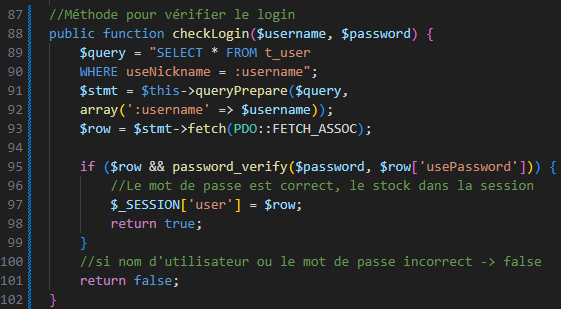


Figure 40 - Fonction checkLogin

### Détails du compte

L’utilisateur a la possibilité, s’il le souhaite, modifier des informations sur son compte s’il le souhaite : un nouvel onglet est disponible grâce à la connexion.

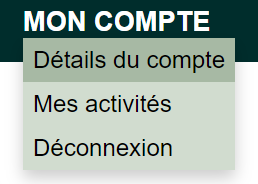


Figure 41 - onglet "Détails du compte"

Arrivé sur la page « userDetails.php », toutes les informations, même le statut « d’Etudiant », est spécifié et la possibilité d’étoffer son profil (ajouter son e-mail ou son genre) est possible. Si l’utilisateur était un enseignant, le titre « Enseignant » indiqué à la place.



Figure 42 - Informations générales

Une fois appuyé sur le bouton « modifié », les champs deviennent modifiables.

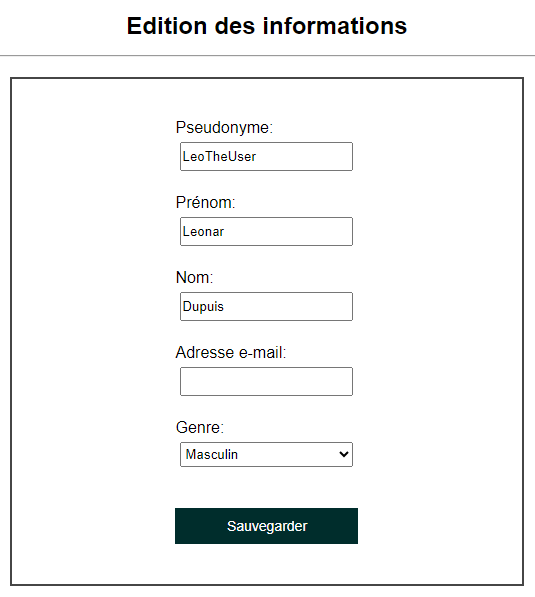


Figure 43 - Modification infos utilisateur

Arrivé sur la page « userDetails.php », toutes les informations, même le statut « d’Etudiant », est spécifié et la possibilité d’étoffer son profil (ajouter son e-mail ou son genre) est possible.

La saisie des champs restent tout de même stricte : Si l’utilisateur insère des chiffres dans son prénom, par exemple, un message d’erreur survient :

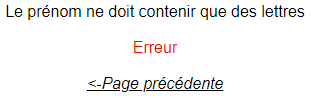


Figure 44 - Erreur champ utilisateur

Pour mettre à jour les données de l’utilisateur, la page « userDetailsCheck.php » dans mon controllers s’occupe d’appeler la méthode updateUserInfo.

A partir de l’identifiant de l’utilisateur, on utilise la requête UPDATE pour effectuer une mise à jour avec les nouvelles informations entrées par l’utilisateur précédemment. Si la méthode retourne ‘true’, la mise à jour a réussie, sinon un message d’erreur est montré à l’utilisateur.

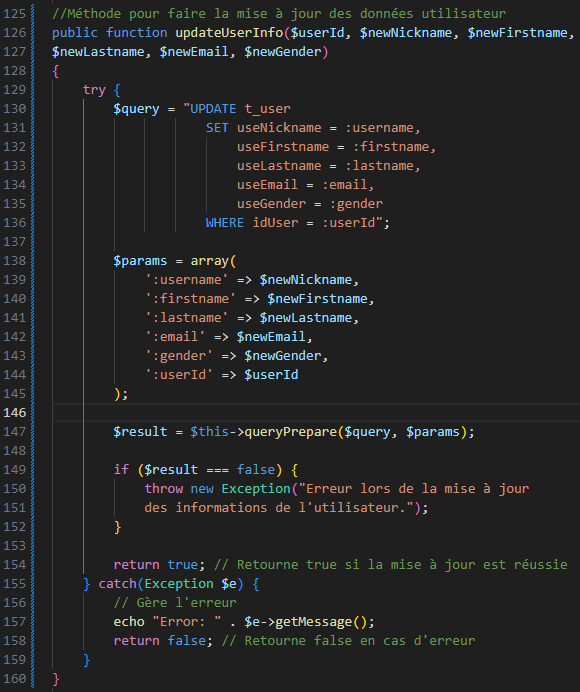


Figure 45 - méthode updateUserInfo

### Opérations CRUD

Les opérations de base « CRUD » concernent ici les interactions entre les activités et les utilisateurs, et plus précisément les enseignants.

En prenant un compte utilisateur enseignant, on peut tout d’abord créer une activité en accédant à l’onglet « Mes activités » en haut à droite de la page (myActivities.php).

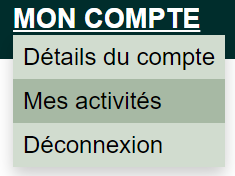


Figure 46 - onglet "Mes activités"

Un message apparaît ici, que ce soit en tant qu’élève ou enseignant, que l’utilisateur est affilié à aucune activité. L’enseignant ici a le droit de créer une activité :

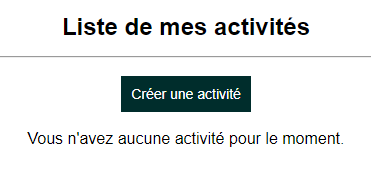


Figure 47 - Liste de mes activités (vide)

En cliquant sur le bouton, un formulaire permettant de créer l’activité apparaît.

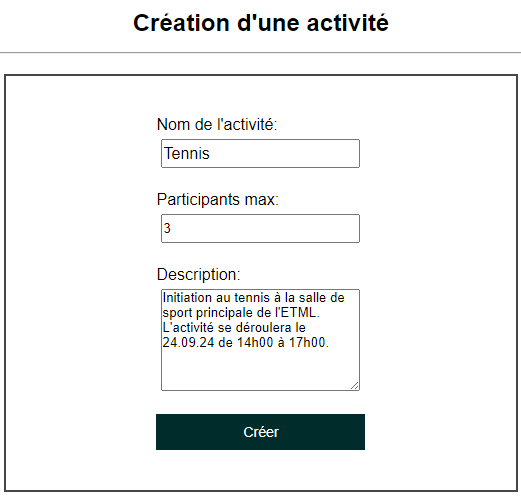


Figure 48 - Création d'une activité

Si l’enseignant oublie d’indiquer une information, un message lui dira de remplir tous les champs. Dans un autre cas, le nombre de participants est négatif, un message survient

|  |  |
| --- | --- |
| Figure 49 - erreur champs vides | Figure 50 - Erreur champ négatif |

Une fois le bouton « créé » appuyé à la fin du formulaire, la méthode createActivity dans la partie « models » va insérer une nouvelle activité dans la table t\_activity avec les informations fournies : titre, description et capacité comme vu dans le formulaire précédent.

L’association est également effectuée entre le créateur de l’activité (l’enseignant) et cette activité dans la table t\_participer.

La méthode retourne l’ID de l’activité créée si réussie, sinon la méthode retourne false.

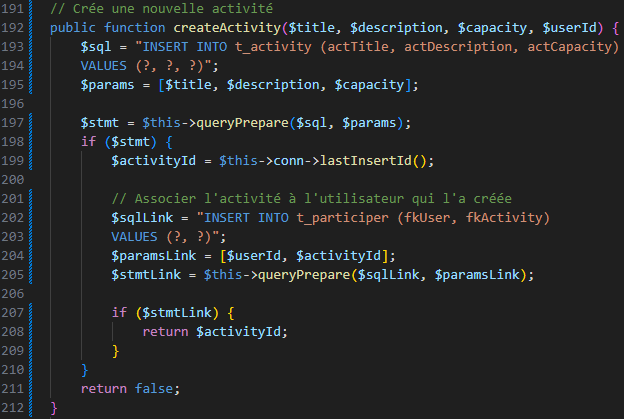


Figure 51 - méthode createActivity

Une fois de retour sur sa page personnelle, l’utilisateur peut voir la nouvelle activité avec plusieurs actions disponibles (un message lui indique si l’activité a bien été créée ou non).



Figure 52 - Activité créée

On vérifie dans la base de données que l’activité a bien été créée. Pour ce faire, on consulte les nouveaux champs dans les colonnes des tables *t\_activity et t\_participer.*

|  |  |
| --- | --- |
| Figure 53 - t\_activity | Figure 54 - t\_participer |

Ce tableau est rempli grâce à une fonctionnalité permettant de récupérer les activités pour un utilisateur grâce à son ID (**getActivitesForUser**).

Dans le cas où l’on souhaite apporter des modifications à notre activité pour une quelconque raison, un bouton « modifier » est disponible pour l’enseignant. S’il clique dessus, un formulaire prérempli avec les informations précédentes sont affichées, avec la possibilité bien sûr de modifier les champs.



Figure 55 - Modification d'une activité

Les conditions pour l’admission de ce formulaire sont les mêmes que pour la création, mais avec une subtilité cette-fois ci.

Si l’enseignant souhaite baisser le nombre de participants en dessous du nombre inscris, cela ne va pas être possible et un message va s’afficher.

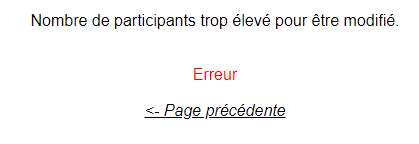


Figure 56 - Erreur nombre de participants élevé

Pour gérer cette erreur, on appelle la méthode **getParticipantCount**.

Celle-ci compte le nombre d’élèves participant à une activité.

La requête utilise la fonction COUNT(\*) et va chercher le nombre d’utilisateurs de type ‘S’ (useType = ‘S’), donc élève, pour une activité donnée.

En effet, l’enseignant étant aussi un participant, je ne veux pas qu’il soit comptabilisé dedans. En prenant le cas de figure où une activité possède une capacité de 5 places, si on ajoute l’enseignant, alors celle-ci en possède en réalité 4.

La fonction retourne finalement le nombre de participants, et si aucun résultat n’est trouvé : 0.

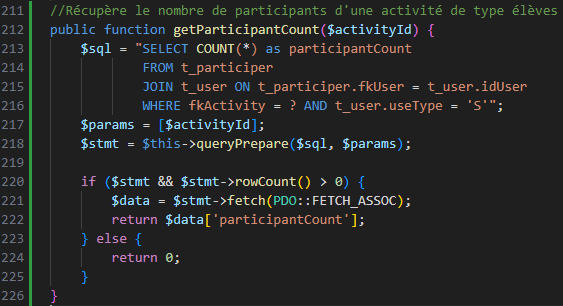


Figure 57 - méthode getParticipantCount

Finalement, la capacité de l’activité sera comparée avec le nombre de participants. Si elle est plus petite, alors la modification est impossible.



Figure 58 - Vérification nombre de participants

Si toutes les conditions sont réunies pour envoyer le formulaire, alors on appelle la méthode **updateActivity** pour appliquer les modifications dans la base de données.

Celle-ci permet la mise à jour des informations d’une activité (titre, description, capacité) sélectionnée par l’enseignant en mettant à jour les colonnes ‘*actTitle’*, ‘*actDescription’* et ‘*actCapacity’* de la table *t\_activity*. L’activité est reconnue grâce à $activityId.

Le résultat est ‘true’ si la mise à jour a réussi, sinon ‘false’.

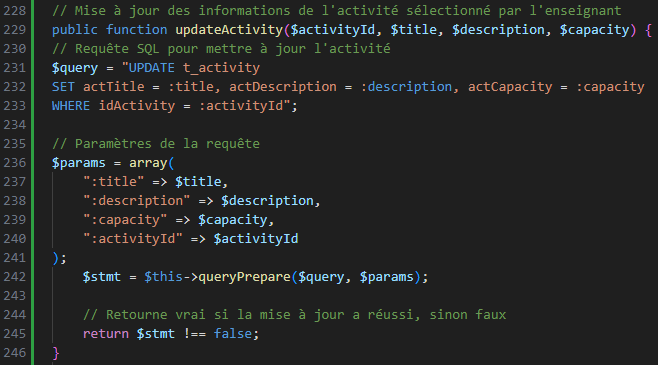


Figure 59 - méthode updateActivity

L’utilisateur sera ensuite redirigé sur la page de ses activités, avec un message lui signalant si l’activité a bien été modifié ou non.

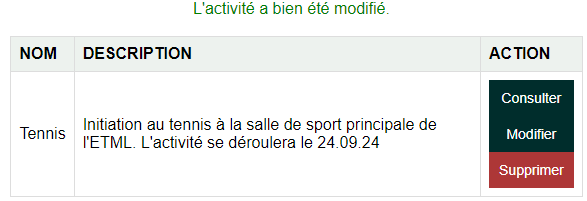


Figure 60 - Activité modifiée

Une des 4 opérations CRUD indispensable ici est la suppression d’une activité. Toujours sur le tableau précédent, en cliquant sur le bouton « Supprimer », un pop-up\* apparaît sur l’écran de l’enseignant.

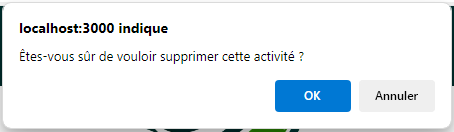


Figure 61 - pop-up supprimer l'activité

Une des 4 opérations CRUD indispensable ici est la suppression d’une activité. Toujours sur le tableau précédent, en cliquant sur le bouton « Supprimer », une boite de dialogue\* apparaît sur l’écran de l’enseignant lui demandant de confirmer la suppression de l’activité.

Du javascript est utilisé ici avec une fonction appelée **confirmDelete** pour supprimer l’activité via une fenêtre pop-up. Lorsque l’utilisateur clique sur ‘confirmer’, l’utilisation du ‘fetch’ permet d’envoyer la requête ‘DELETE ‘ vers la partie controllers (*deleteActivity.php*) avec l’ID de l’activité à supprimer. Si la réponse est correcte (‘*response.ok’*), la page est rechargée pour actualiser l’affichage, sinon il y’a une erreur.

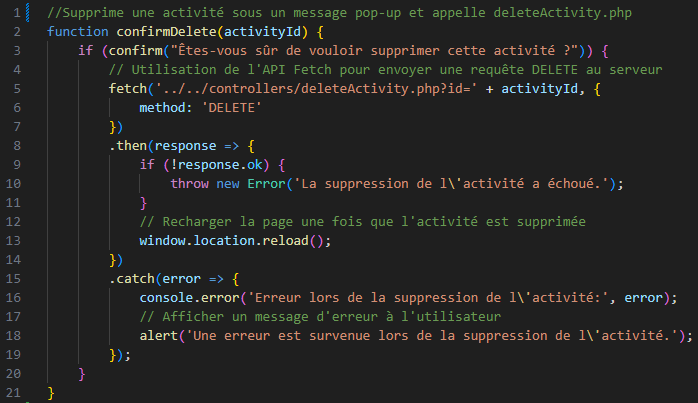


Figure 62 - script.js

Pour revenir sur le côté PHP, ma page *deleteActivity.php* va servir à appeler la méthode **deleteActivity**.

Cette méthode a pour objectif de supprimer une activité spécifique grâce à la requête DELETE qui va supprimer toutes les lignes de la table *‘t\_participer’* où la colonne *‘fkActivity’* correspond à l’identifiant de l’activité. Cela évite tout scénario de bug où un participant est encore associé à une activité qui n’existe plus.

Par la suite, la requête DELETE est utilisée pour supprimer la ligne de la table *‘t\_activity’* où la colonne *‘idActivity’* correspond à l’identifiant de l’activité.

Pour finir, si les deux suppressions ont réussi, la méthode retourne ‘true’. En revanche, si l’une des suppressions a échoué, la méthode retourne ‘false’.

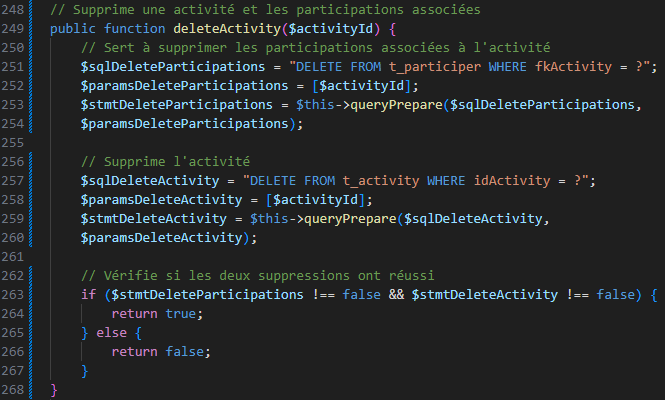


Figure 63 - méthode deleteActivity

|  |  |
| --- | --- |
| Figure 64 - Erreur lors de la suppression | Figure 65 - Activité supprimée |

La dernière opération CRUD intégrée sur le site permet la consultation des détails d’une activité. Pour accéder aux détails d’une activité, il faut juste cliquer sur le bouton « Consulter » à côté de l’activité souhaité, par exemple depuis notre profil « Mes activités ».



Figure 66 - Détails de l'activité

Des fonctionnalités mineures comme la récupération des détails d’une activité via l’ID de celle-ci (**getActivityById**), ou encore la récupération des informations d’un enseignant via le même paramètre (**getActivityOrganizer**) sont utilisées. En revanche, cette page contient une particularité, plus précisément concernant l’affichage du statut de l’activité.

La méthode utilisée ici vérifie si la capacité maximale d'une activité est atteinte en comparant le nombre actuel de participants de type élève (grâce à une jointure) avec la capacité maximale définie de base.

La requête COUNT compte le nombre actuel de participants afin d’effectuer une condition vérifiant que *‘actCapacity’* est supérieur à *‘currentParticipants’*, sinon, on retourne ‘false’.

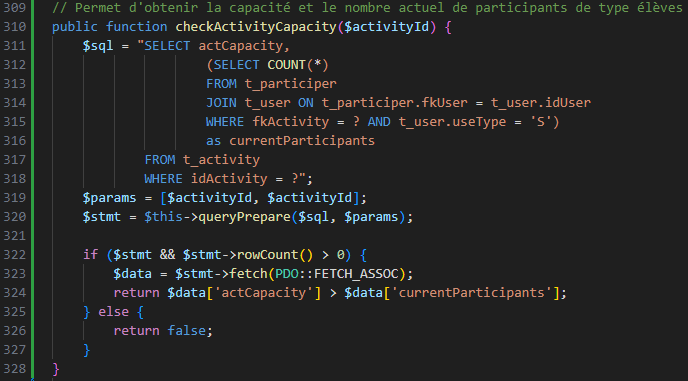


Figure 67 - méthode checkActivityCapacity

L’inscription n’étant pas encore réalisé à ce stade, on crée et on insert les élèves dans une des activités depuis la base de données dans l’objectif de vérifier que le statut de l’activité change. On voit ici que chaque utilisateur est différent, mais associé à la même activité dans la table *t\_participer*. L’enseignant ici n’est pas comptabilisé grâce à la jointure.

|  |  |
| --- | --- |
| Figure 68 - t\_participer - 4 élèves | Figure 69 - activité indisponible |

### Liste des participants de l’activité

Sur la page affichant les détails d’une activité se trouve la liste des participants. Celle-ci, bien que visible pour tous les utilisateurs en bas de la page, et est modifiable au besoin par l’organisateur (et non par les élèves) en ajoutant ou supprimant des participants.



Figure 70 - Liste des participants

Pour ajouter des participants, on recherche le nom, prénom ou pseudo d’un utilisateur en fonction des lettres de saisies. L’utilisateur va apparaître avec un bouton pour l’ajouter à l’activité.

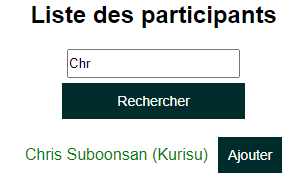


Figure 71 - Recherche d'utilisateur

Quelques méthodes sont utilisées ici pour que cette partie de l’application fonctionne. L’une récupère tous les utilisateurs inscris à la même activité pour les afficher dans le tableau (**getParticipantsForActivity**) et une autre supprime tout simplement un utilisateur d’une activité en fonction de l’ID de l’utilisateur et de l’activité via une requête DELETE FROM (**removeParticipantFromActivity**). Les méthodes suivantes sont cependant assez différentes.

Dans cette méthode, les colonnes ‘*idUser’*, ‘*useFirstname’*, ‘*useLastname’* et ‘*useNickname’* de la table ‘*t\_user’* sont sélectionnées dans l’objectif de vérifier, avec WHERE, si une de ces colonnes contient un terme correspondant au paramètre $searchTerm. Le LIKE avec les ‘%’ permet de rechercher des correspondances partielles dans ces colonnes. Par exemple, le champ « *chr* » a été tapé dans la barre de recherche précédente et la méthode renvoie le résultat « *Chris* ».

Pour finir, le résultat de la méthode est retourné comme la méthode précédente (**getParticipantsForActivity**).

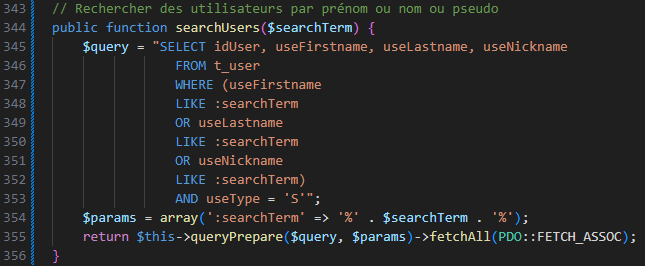


Figure 72 - méthode searchUsers

La troisième méthode utilisée ici sert à ajouter un participant à une activité.

On insère simplement un nouvel enregistrement dans la table *t\_participer*, dans les colonnes ‘*fkUser’* et ‘*fkActivity’*, avec les paramètres $userId et $activityId.

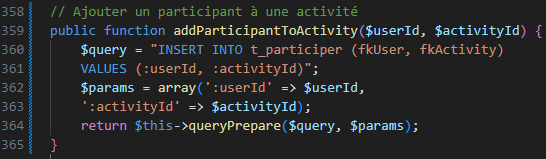


Figure 73 - méthode addParticipantToActivity

### Liste des activités & Inscription

Depuis n’importe quel page du site, la liste des activités est accessible sur la barre de navigation en haut à gauche.



Figure 74 - Onglet "Liste des activités"

La page suivante s’affiche, dressant une liste de toutes les activités existantes grâce à un foreach. Si l’utilisateur est un élève, la possibilité de s’inscrire à l’activité s’offre à lui, contrairement à l’enseignant.

La page ici reprend les méthodes suivantes : **getAllActivities**, **getActivityOrganizer**, **checkActivityCapacity** et **addParticipantToActivity**.



Figure 75 - Liste des activités

L’inscription fonctionne uniquement s’il reste de la place pour l’activité, ou que l’utilisateur n’est bien entendu pas encore inscrit à celle-ci.

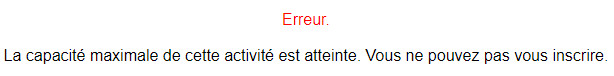


Figure 76 - Activité maximale atteinte

|  |  |
| --- | --- |
| Figure 77 - Inscription déjà effectuée | Figure 78 - Inscription réussie |

Ces messages sont affichés en fonction des résultats obtenus des méthodes avec des conditions gérés dans la partie controllers de la page (subscribe.php).

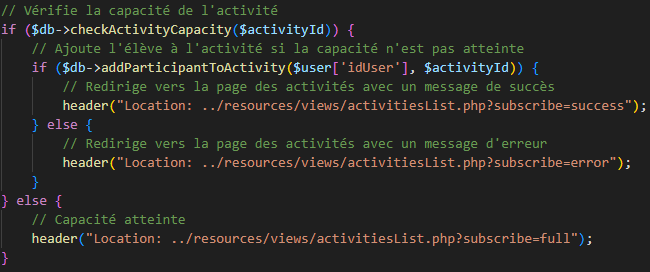


Figure 79 - Conditions inscription

### Consulter un profil utilisateur

La dernière fonctionnalité de ce site consiste à pouvoir consulter le profil utilisateur de quelqu’un d’autre, incluant des informations concernant les activités auxquelles il est affilié. Pour un élève, cela signifie les activités auxquelles il participe, et pour un enseignant, cela comprend les activités qu’il organise.

Les profils sont consultables via des hyperliens\*, visibles lorsqu’un nom, prénom ou un pseudo est souligné sur le site.

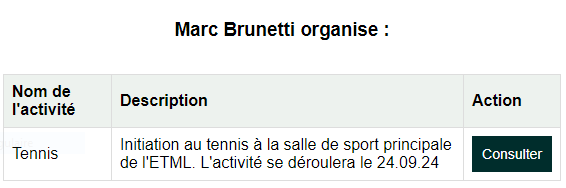


Figure 80 - Profil d'activités d'un enseignant

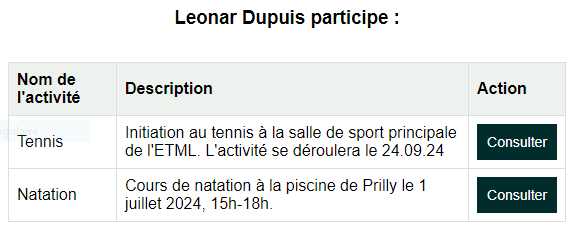


Figure 81 - Profil d'activités d'un élève

L’implémentation d’une méthode appelée « **getUserById** » permet ici de récupérer les informations d’un utilisateur à partir de son ID grâce à une requête SELECT.

## Modifications

Historique des modifications demandées (ou nécessaires) aux spécifications détaillées.

Date, raison, description, etc.

# Tests

## Dossier des tests

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Fonctionnalité** | **Procédure de test** | **Résultat attendu** | **Résultat obtenu** | **Correction** |
| **Authentification** | *Cas de test 1*   1. Accéder à la page de connexion 2. Saisir un nom d’utilisateur et un mot de passe valides/existants 3. Cliquer sur le bouton de connexion 4. Vérifier la redirection vers la page d’accueil et une page protégée 5. Vérifier que l’utilisateur est bien authentifié (session)   *Cas de test 2*  Même procédure, en saisissant un nom d’utilisateur ou un mot de passe incorrect  *Cas de test 3*   1. Accéder à une page protégée sans se connecter 2. Vérifier la redirection vers la page de connexion | *Cas de test 1*  Redirection vers la page d’accueil avec un nouvel accès au compte en haut à droite, informations de l’utilisateur stocké dans la session.  *Cas de test 2*  Message indiquant que la connexion a échouée. Aucune session active.  *Cas de test 3*  Redirection vers la page de connexion. Aucune session active. | *Cas de test 1*  **Réussi**.  Nom d’utilisateur :  «**LeoTheUser**»  Mot de passe :  « **Mnbvcxy0**»  La session est active et l’accès au compte dans la barre de navigation fonctionne  *Cas de test 2*  **Réussi**.  Nom d’utilisateur :  «**LeoTheUser**»  Mot de passe :  «**Fauxmotdepasse**»  La session n’est pas active et un message d’erreur survient.  *Cas de test 3*  **Réussi**.  Tentative d’accès à la liste des activités -> redirection vers la page de connexion. | - |
| **Fonctionnalité** | **Procédure de test** | **Résultat attendu** | **Résultat obtenu** | **Correction** |
| **Opérations**  **CRUD** | *Cas de test 1*   1. Accéder à la page « Mes activités » pour créer une nouvelle activité 2. Remplir les champs du formulaire (test champs vides) 3. Cliquer sur le bouton « Créer » 4. Vérifier la redirection vers la liste des activités avec un message de succès. 5. Vérifier l’ajout de l’activité dans la base de données   *Cas de test 2*  Vérifier le contenu détaillé d’une activité en appuyant sur « Consulter »  *Cas de test 3*  Même procédure que le cas de test 1, mais pour modifier une activité  *Cas de test 4*   1. Appuyer sur le bouton « supprimer à côté d’une activité, puis confirmer dans la boite de dialogue. 2. Vérifier la redirection avec un message de succès. 3. Vérifier la suppression de l’activité dans la base de données | *Cas de test 1*  Redirection vers la liste des activités avec un message de succès. Nouvelle activité ajoutée dans la table ‘*t\_activity’*  *Cas de test 2*  Les détails de l'activité sélectionnée sont affichés de manière correcte, avec le statut géré selon le nombre d’inscrits.  *Cas de test 3*  Redirection vers la liste des activités avec un message de succès. MàJ\* des champs dans la table ‘*t\_activity*’  *Cas de test 4*  Redirection vers la liste des activités avec un message de succès. L’activité est supprimée de la table ‘*t\_activity’* | *Cas de test 1*  **Echec.**  Les champs vides sont gérés avec un message d’erreur. A l’inverse, un message « succès » ainsi qu’une nouvelle entrée est visible dans *‘t\_activity’.*  Cependant*,* un problème apparaîtlors de la création d'une deuxième activité : celle-ci reprend les mêmes informations que la première.  Lors de la saisi du formulaire, un champ négatif (*ex. : -1*) pour les participants provoque un crash du localhost  *Cas de test 2*  **Réussi**.  Les détails et le statut affichés sont corrects  *Cas de test 3*  **Echec**.  Message de succès et mis à jour de l’entrée dans ‘*t\_activity’*, mais si la capacité est inférieure au nombre de participants inscrits, localhost crash  Cas de test 4  **Réussi.**  Message de succès et entrée supprimée de *‘t\_activity’* | *Cas de test 1*  **Réussi**.  Ajout de la fonction unset() pour casser le dernier élément référencé dans mon tableau, ainsi que l’ajout d’une condition n’acceptant pas les champs négatifs  *Cas de test 3*  **Réussi.**  Création d’une méthode appelée **getParticipantCount** permettant de récupérer le nombre de participants, avec une condition permettant de gérer l’erreur si la capacité est inférieure au nouveau nombre max. de participants. |
| **Fonctionnalité** | **Procédure de test** | **Résultat attendu** | **Résultat obtenu** | **Correction** |
| **Modifier la liste des participants** | *Cas de test 1*   1. Accéder à la page de la liste des activités 2. Rechercher un élève pour l’ajouter à l’activité 3. Appuyer sur le bouton pour l’ajouter comme participant 4. Vérifier la mise à jour de la liste des participants 5. Vérifier la nouvelle entrée dans la table ‘*t\_participer’*   *Cas de test 2*  Même procédure, mais en essayant d’ajouter un élève déjà inscris à la même activité  *Cas de test 3*   1. Supprimer un élève dans la liste des participants en appuyant sur le bouton « Supprimer » à côté de son pseudo. 2. Vérifier la mise à jour dans la liste des participants 3. Vérifier qu’il n’est plus dans la table *‘t\_participer’* | *Cas de test 1*  Redirection vers la liste des activités avec un message de succès. La liste des participants est MàJ et une nouvelle entrée est visible dans ‘*t\_participer’*  *Cas de test 2*  Ajout non effectué, rien n’est modifié ou ajouté dans la liste des participants ou dans la base de données  *Cas de test 3*  Redirection vers la liste des activités avec un message de succès. La liste des participants est MàJ et l’entrée est supprimée de ‘*t\_participer’*. | *Cas de test 1*  **Réussi.**  Saisi de la recherche :  « **chr** »  Utilisateur trouvé :  «**Chris Suboonsan**»  L’utilisateur est ajouté et apparaît dans la liste des participants avec un message de succès. Une nouvelle entrée est visible dans *‘t\_participer’*.  *Cas de test 2*  **Réussi.**  Aucune information n’est modifiée ou ajoutée dans la liste des participants ou la base de données.  *Cas de test 3*  **Réussi.**  Le participant est supprimé et n’apparaît plus dans la liste des participants, et l’entrée correspondante est supprimée de la table *‘t\_participer’*. | - |
| **Fonctionnalité** | **Procédure de test** | **Résultat attendu** | **Résultat obtenu** | **Correction** |
| **Consulter les activités**  **&**  **S’inscrire à une activité** | *Cas de test 1*   1. Accéder à la page « liste des activités » en vérifiant que toutes les activités sont bien affichées 2. Sélectionner une activité avec des places disponibles 3. Cliquer sur le bouton d’inscription 4. Vérifier le message de succès 5. Confirmer l’ajout de l’inscription dans la base de données   *Cas de test 2*  Même procédure, mais en sélectionnant une place indisponible  *Cas de test 3*  Même procédure, en revanche l’utilisateur est déjà inscrit à l’activité | *Cas de test 1*  Toutes les activités sont visibles. Message de succès : Redirection vers la page initiale avec le paramètre ‘subscribe=success’.  Nouvelle entrée dans la table *‘t\_participer’* dans la base de données  *Cas de test 2*  Message de d’erreur : Redirection vers la page initiale avec le paramètre ‘subscribe=full’.  Aucune nouvelle entrée dans la base de données  *Cas de test 3*  Message de d’erreur : Redirection vers la page initiale avec le paramètre ‘subscribe=error’.  Aucune nouvelle entrée dans la base de données | *Cas de test 1*  **Réussi**.  Message de succès visible pour l’utilisateur et nouvelle entrée dans ‘*fkUser’* et ‘*fkActivity’*  *Cas de test 2*  **Réussi**.  Message d’erreur signalant que l’activité est déjà remplie. Aucune nouvelle entrée dans la base de données  *Cas de test 3*  **Réussi**.  Message d’erreur signalant que l’utilisateur est déjà inscrit à l’activité. Aucune nouvelle entrée dans la base de données | - |
| **Fonctionnalité** | **Procédure de test** | **Résultat attendu** | **Résultat obtenu** | **Correction** |
| **Consulter un profil utilisateur** | *Cas de test 1*   1. Se connecter à l’application en tant qu’utilisateur (enseignant ou élève). 2. Accéder à sa page de profil 3. Modifier ses informations du profil (et test champs vides) 4. Enregistrer les informations 5. Vérifier la MàJ sur son profil 6. Vérifier la MàJ dans la base de données   *Cas de test 2*   1. Sur une des pages du site, cliquer sur un prénom, nom ou pseudo en hyperlien d’un utilisateur afin d’accéder à son profil. 2. Vérifier si les informations du profil sont correctes et la différence d’affichage des activités entre un compte élève et un compte enseignant. Vérifier également la gestion de l’affichage en cas d’affiliation à aucune activité. | *Cas de test 1*  Un message de succès apparaît indiquant que les informations du profil ont bien été MàJ.  Les informations modifiées sont MàJ dans la table *‘t\_user’*  *Cas de test 2*  Les informations du profil de l’utilisateur consulté sont affichées correctement.  La différence d’affichage entre les activités enseignant/élève est correct. L’affichage est géré avec un message en cas d’absence d’affiliation des activités. | *Cas de test 1*  **Réussi.**  Les informations du profil sont mises à jour, et la mise à jour n’est pas effectué si les champs sont laissés vides dans le formulaire.  *Cas de test 2*  **Echec**.  La consultation des informations d'autres utilisateurs est possible. Les activités sont notées « organise » pour un enseignant et « participe » pour un élève.  L'absence d'affiliation à une activité est signalée par un message.  En revanche, la gestion des erreurs ne couvre pas le cas où un utilisateur n'est pas trouvé (ID non trouvé), ce qui provoque un crash de la page localhost. | *Cas de test 2*  **Réussi.**  Ajout d’un isset dans la page controllers vérifiant si l’ID est donné, sinon, affiche un message signalant que l’utilisateur n'est pas trouvé. |

# Conclusion

## Bilan des fonctionnalités demandées

Il s’agit de reprendre point par point les fonctionnalités décrites dans les spécifications de départ et de définir si elles sont atteintes ou pas, et pourquoi.

Si ce n’est pas le cas, estimer en « % » ou en « temps supplémentaire » le travail qu’il reste à accomplir pour terminer le tout.

## Bilan de la planification

Distinguer et expliquer les tâches qui ont généré des retards ou de l'avance dans la gestion du projet. Indiquer les différence entre les planifications initiales et détaillées avec le journal de travail.

## Bilan personnel

Si c’était à refaire:

Qu’est-ce qu’il faudrait garder ? Les plus et les moins ?

Qu’est-ce qu’il faudrait gérer, réaliser ou traiter différemment ?

Qu’est que ce projet m’a appris ?

Suite à donner, améliorations souhaitables, …

Remerciements, signature, etc.

# Divers

## Journal de travail

Date, activité (description qui permet de reproduire le cheminement du projet), durée, liens et références sur des documents externes. Lorsqu’une activité de recherches a été entreprise, il convient d’énumérer ce qui a été trouvé, avec les références.

## Bibliographie

Références des livres, revues et publications utilisés durant le projet.

## Webographie

Références des sites Internet consultés durant le projet.

# Annexes

## Résumé du rapport du TPI / version succincte de la doc.

-Situation de départ

-Mise en œuvre

-Résultats

## Glossaire